Closed-Loop Agentic Reinforcement Learning for Vision-Language Driving Agents

Anonymous authors
Paper under double-blind review

Abstract

Vision-language driving policies commonly reason through a fixed chain of structured questions before acting, asking the same questions at every frame. On Bench2Drive-VL, a closed-loop benchmark that scores driving and reasoning on the same episodes, a model answering this chain drives worse than one answering nothing: which questions a frame warrants is itself a decision, and a fixed chain never makes it. Adaptive-reasoning methods do make it, but optimise the accuracy or cost of the answers where they are produced; the value of a question is realised only in the trajectory that follows. We formalise reasoning-chain selection as a closed-loop Markov decision process, which makes this mismatch explicit. On this basis we introduce DynaChain, an agentic driving policy that decides at every frame which questions to ask before committing to an action. We train it with closed-loop agentic reinforcement learning, under a hybrid advantage that combines the episode's driving outcome with a dense per-step reward derived from the benchmark's scoring rule. DynaChain raises the Driving Score of its base model from 58.93 to 79.15 while asking 2.52 questions per decision instead of seven, and outperforms two adaptive-reasoning methods. Our analysis shows that the gain comes from which questions the agent asks rather than how accurately it answers them, and that it requires both learned selection and closed-loop training. We will release our source code.

1 Introduction

Vision-language models have made it possible for a driving policy to reason in language before it acts (Sima et al., 2024; Nie et al., 2024; Xu et al., 2024; Tian et al., 2024), rather than mapping sensor input to control directly (Chitta et al., 2023; Hu et al., 2023; Jiang et al., 2023). A common way to elicit this reasoning, in the spirit of chain-of-thought prompting (Wei et al., 2022), is a chain of structured questions answered in a fixed order at every frame, which makes the policy's reasoning explicit and easy to inspect. The closed-loop benchmark we build on (Jia et al., 2026) adopts this design and scores the driving outcome and the intermediate answers on the same episodes.

That design does not work. On this benchmark a camera-input model answering the entire chain attains a Driving Score of 42.24, against 58.93 for the same model asked nothing at all. In open-loop question answering such a result would be hard to explain: an extra question costs computation and a wrong answer costs its own score, but neither should make the rest of the episode worse.

Closed-loop control removes that guarantee. A fixed chain answers every question at every frame, including questions the scene does not raise, and all of these answers feed into the control decision. Whether an answer helps depends on the scene, which a fixed chain never considers, and the longer the chain, the more answers there are that can mislead. The benchmark's negative result is not evidence against reasoning before acting, but against reasoning identically everywhere: *which* questions a frame warrants is itself a decision, and a fixed chain never makes it (Figure 1).

Once selection is a decision, what should train it? Adaptive-reasoning methods optimise the accuracy of each answer where it is produced. But the value of asking a question is not whether its answer is correct; it is the difference the answer makes to the rest of the drive, which may show only seconds and many decisions later. Answer accuracy thus measures a different quantity, not a noisier version of the right one. The two agree only under an assumption the benchmark's own result refutes (§3.1), so the selection policy must be trained on the closed-loop outcome.

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.

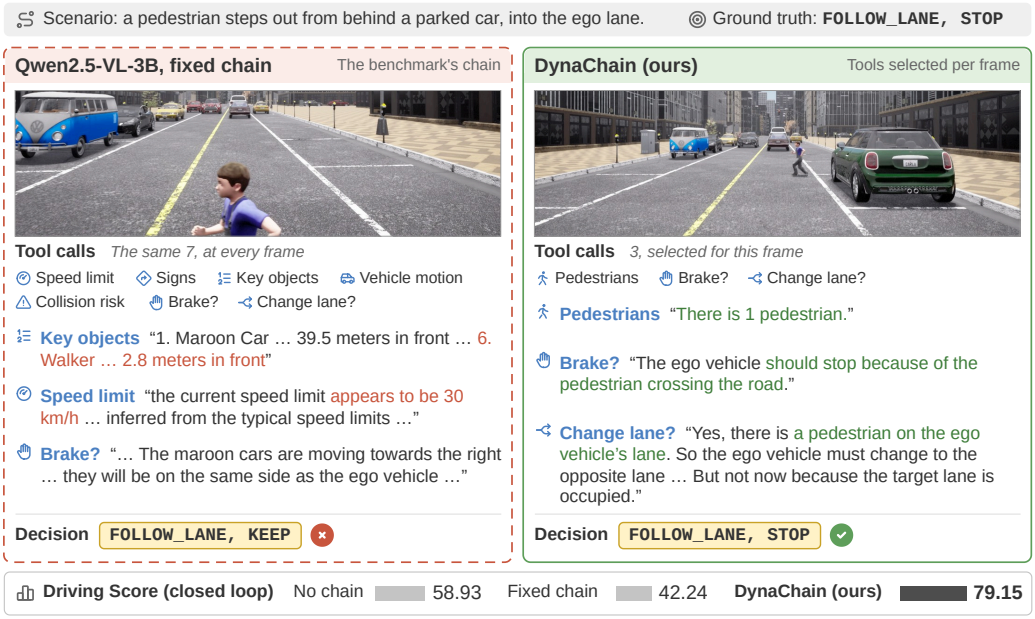

Figure 1: One scenario, two policies. A pedestrian steps out from behind a parked car; the ground-truth decision is to stop. The base model with the benchmark's fixed chain asks the same seven questions it asks at every frame, answers them about the wrong things, and decides `FOLLOW_LANE, KEEP`; DynaChain asks three, all answered about the pedestrian, and decides `FOLLOW_LANE, STOP`.

We address this with DynaChain, an agentic vision-language driving policy that learns, frame by frame and in closed loop, which questions to ask. Within a frame the agent alternates between choosing a question and answering it; the turn that stops asking also gives the driving decision, which the benchmark's own controller executes unchanged. We train the selection policy on the closed-loop Driving Score, with an advantage that combines the sparse route-level outcome with a dense per-step reward derived from the benchmark's scoring rule rather than set by hand. DynaChain asks 2.52 questions per decision instead of seven, and reaches a Driving Score of 79.15, against 42.24 for the fixed chain and 58.93 for no chain.

Our main contributions are summarised as follows:

1. We formalise reasoning-chain selection as a closed-loop Markov decision process, in which asking a question and stopping to act are actions whose value shows only in how the environment responds, and we state the assumption under which an open-loop objective would agree with it (§3.1).
2. We build DynaChain, an agentic driving policy whose reasoning chain is a sequence of actions rather than a fixed graph: at each turn it either asks one question, answered by an isolated call that does not see the rest of the chain, or commits. The controller is never trained (§3.2).
3. We design a closed-loop agentic reinforcement-learning procedure for this setting. It extends GRPO (Shao et al., 2024b) with a hybrid advantage: the group-relative return of the episode plus a dense per-step term whose infraction weights are *derived* from the benchmark's scoring rule by taking logarithms, and spread back onto the decisions that caused them (§3.3). Ablations show that the gain needs learned rather than merely sparse selection, closed-loop training, and both reward terms (§4.3).

2 Related Work

Vision-language models for autonomous driving. Vision-language models (VLMs) have been applied to driving both as end-to-end planners (Hwang et al., 2025; Renz et al., 2025; Fu et al.,

2025) and as reasoning front-ends that explain a decision before it is made (Tian et al., 2024). A common design elicits it through a predetermined sequence of structured questions: Sima et al. (2024) organise perception, prediction and planning question-answering into an explicit dependency graph, and Nie et al. (2024) scale it to a large video-text corpus; related datasets annotate driving scenes with free-form (Marcu et al., 2024), counterfactual (Wang et al., 2025a) or step-by-step (Ishaq et al., 2025) reasoning, while Xu et al. (2024) instead fine-tune a single model to emit a rationale and a control signal jointly, without a fixed question schema. In all of them the rationale is supervised open-loop and never evaluated against the driving behaviour it induces; Shao et al. (2024a) do drive a language model in closed loop, but from natural-language instructions and without an intermediate rationale. The benchmark on which we build (Jia et al., 2026) removes this separation, scoring the driving outcome and the intermediate answers on the same episodes, and reports the observation that motivates this work: a camera-input model drives substantially worse when it answers the full graph than when it is asked no questions at all. A complementary line applies reinforcement learning in a closed-loop simulator, but optimises the action mapping rather than the reasoning before it: Li et al. (2024) train a planner within a learned latent world model; Shi et al. (2026) learn a residual waypoint policy around the prior of a pretrained policy; and Fu et al. (2026) train a vision-language-action (VLA) model online in closed loop. Where a language rationale exists in these systems at all, it is free-form text, and none of them treats which reasoning steps to take as a decision in its own right; we instead keep the controller frozen and apply reinforcement learning to the composition of the reasoning itself.

Agentic reinforcement learning for vision-language models. Reinforcement learning with outcome rewards elicits extended reasoning in language models (Guo et al., 2025) and teaches them to call tools such as a search engine mid-reasoning (Jin et al., 2025). A growing body of work extends this to VLMs, invoking perceptual operations on their own input as callable tools and appending the returned observations to an interleaved trace: Zheng et al. (2026b) learn to call an image zoom-in operation mid-reasoning, and Su et al. (2025) add a curiosity term that keeps such policies from collapsing onto text-only reasoning. The formulation has antecedents in language models that call tools mid-generation (Yao et al., 2023; Schick et al., 2023) and in decomposed prompting (Khot et al., 2023), which delegates sub-tasks to dedicated prompts much as our agent delegates each question (§3.2), with the distinction that which sub-task to dispatch is here learned. Outside driving, adaptive reasoning is typically cast as deciding whether or how long to think, trading answer accuracy against reasoning cost (Chen et al., 2025; Fang et al., 2025; Lou et al., 2025). Within driving, reinforcement learning has been applied to VLM reasoning and, in several cases, used to render it adaptive: Jiang et al. (2025) reward planning reasoning over discrete meta-actions via GRPO; Zheng et al. (2026a) alternate between text-only reasoning and tool-augmented perception under a cascaded reinforcement-learning policy; Qian et al. (2025) reward tool-call and answer correctness for a fixed external tool library; and Zhou et al. (2025) switch between trajectory-only and chain-of-thought modes under a reward penalising reasoning length. Beyond driving, reinforcement learning has trained agents whose actions change the environment, including web agents (Qi et al., 2025), multi-turn agents in interactive tasks and games (Zhou et al., 2024; Wang et al., 2025b), and agents that revise their strategy from verbal feedback (Shinn et al., 2023), but not, to our knowledge, to decide which reasoning steps to take. In the vision-language methods above the input image does not change when a tool is called, and the reward for a call is settled within the same reasoning trace, so choosing a tool has no effect beyond the answer it supports. The driving methods that do adapt their reasoning make a coarse choice — whether, for how long, or in which of a few modes to reason — and, with the partial exception of Zheng et al. (2026a), also optimise objectives measured where the choice is made. We instead run agentic reinforcement learning *in closed loop*: the agent's decision moves the simulator forward, so the questions asked at one frame shape what the agent sees at the next, and the value of a choice appears only in the drive that follows.

3 Method

3.1 Reasoning-Chain Selection as a Closed-Loop MDP

We now state this decision formally. Two time scales are involved. At each *control step* t the vehicle acts once. Within a control step the agent takes several *selection steps* $m = 0, 1, \ldots$, each deciding what to ask next. The state $s_t^{(m)}$ holds the observation o_t at frame t (the camera images and the

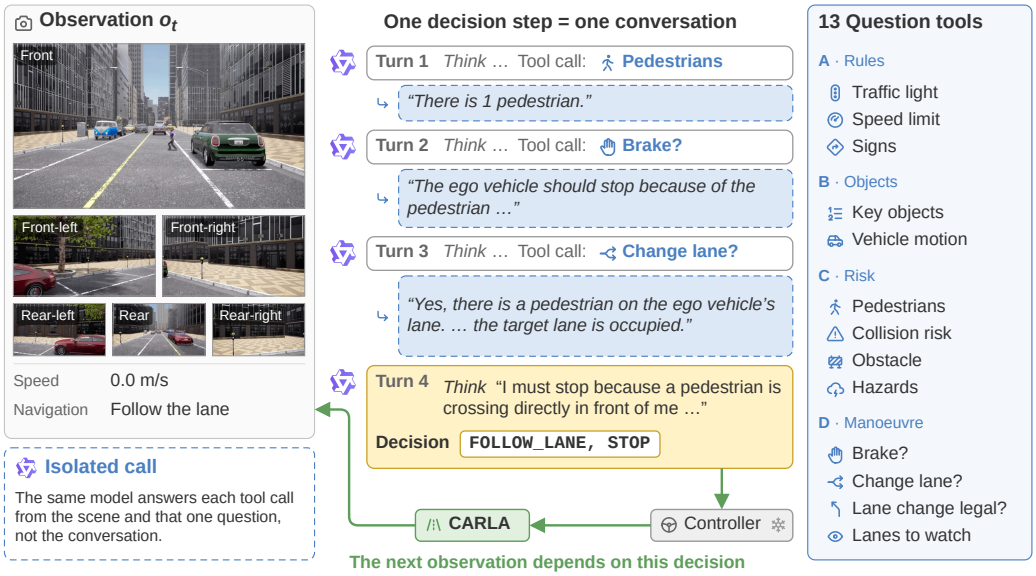

Figure 2: One decision step of DynaChain (Qwen2.5-VL-3B with LoRA), from a real episode. At each turn the agent thinks and either calls one question tool — answered by an isolated call that sees the scene and that question but not the conversation — or ends the step with a decision, which the frozen controller executes. What is asked at one frame shapes the observation at the next.

benchmark's text context) and the m question-answer pairs already collected at this frame; at $m = 0$ there are none.

Let Q be the catalogue of questions the agent can ask (§3.2), and let $q_{t,1}, \ldots, q_{t,m}$ be the questions already asked at frame t. At each selection step the agent either asks a new question or *commits*: it stops asking and outputs the driving decision. The set of actions available in state $s_t^{(m)}$ is therefore

$$\mathcal{A}\big(s_t^{(m)}\big) \;=\; \underbrace{Q \setminus \{q_{t,1}, \ldots, q_{t,m}\}}_{\text{a question not yet asked here}} \;\cup\; \underbrace{\{\text{commit}\}}_{\text{stop asking and act}}, \tag{1}$$

and the selection policy π_θ, i.e. the model with parameters θ, samples one action from this set.

The two actions differ in whether they move the vehicle. Asking does not: the question is answered (by the model itself at evaluation, by a privileged expert during training; §3.3), the answer is added to the state, and the agent chooses again. Committing does: a frozen controller executes the decision, and the simulator returns the next observation o_{t+1}, which depends on that decision. The policy, not a fixed schedule, thus sets how many questions are asked per frame.

This view shows what an open-loop objective misses. Let DS be the Driving Score of the whole episode. The value of asking question q at frame t is how much it raises the expected final score:

$$\Delta V_t(q) \;=\; \mathbb{E}\big[\mathrm{DS} \,\big|\, \text{ask } q \text{ at } t\big] \;-\; \mathbb{E}\big[\mathrm{DS} \,\big|\, \text{skip } q \text{ at } t\big]. \tag{2}$$

Adaptive-reasoning methods instead optimise a quantity measured at frame t itself: how accurate the answers a_t given there are, or how much they cost. The two objectives agree only if the Driving Score grows with answer quality, $\mathrm{DS} = \Phi\big(\sum_t \mathrm{Acc}(a_t)\big)$ for some increasing function Φ. The benchmark's own result contradicts this: the fixed chain bases each decision on more answers, of no lower accuracy, yet drives worse. Answer accuracy is therefore not a noisy estimate of equation 2 but a different quantity, so we train selection on the closed-loop outcome itself.

3.2 The DynaChain Agent

A control step is one conversation (Figure 2). At each turn the agent sees the observation and the earlier turns, reasons briefly inside `<think>...</think>`, and outputs one of two actions:

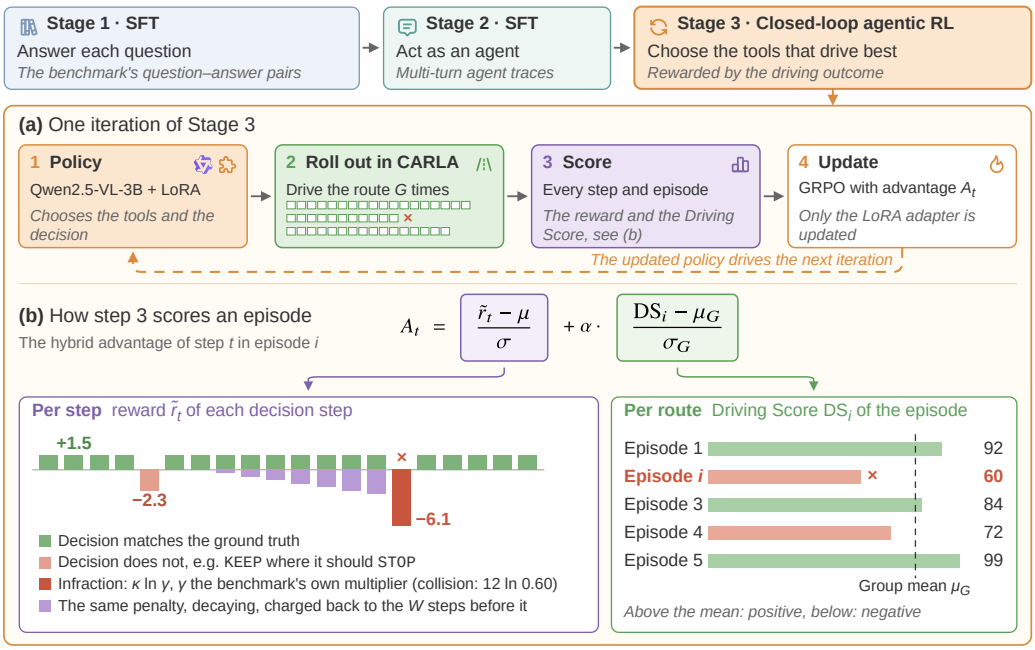

Figure 3: Training. Stage 1 teaches the model to answer each question, Stage 2 to act as an agent from verifier-pruned teacher traces, and Stage 3 which tools to call, by rewarding the driving outcome in CARLA. **(a)** One iteration of Stage 3. **(b)** The hybrid advantage of equation 6, on a schematic episode carrying the values of the actual reward table.

`<tool>NAME</tool>` asks one question, and `<answer>...</answer>` commits: it gives the driving decision and ends the step. There is no separate decision stage; the turn that stops asking is the turn that acts. Its `<think>` text explains the chosen manoeuvre, and the benchmark scores it as one of its questions. The agent sets the number of turns, up to a cap that is rarely reached.

Each tool is one question from the benchmark's annotation schema, answered by the same model with a prompt written for that question. Two properties matter. First, the benchmark, not us, defines the questions, which cover traffic rules, objects, risk and manoeuvres, so the policy cannot profit from wording chosen to suit it. Second, the privileged expert can answer every question, so supervision exists at every intermediate step (§3.3).

Answering is isolated. Each answer is produced from the observation and its own question only, not from the conversation, so no answer can change the premise of another question. The answers are combined in one place: the committing turn, which reads them all. An irrelevant answer can still mislead the decision, but it cannot spread to later questions.

3.3 Closed-Loop Reinforcement Learning

Training stages. Training has three stages (Figure 3); only the third uses driving outcomes. Stage 1 fine-tunes the model to answer each question on its own, supervised by expert annotations, so that asking returns a useful answer. Stage 2 teaches the agent format: a teacher model proposes which questions a frame needs, a verifier removes unnecessary ones, and the resulting traces are used as supervision. Without it, the policy cannot yet commit in the right format, and its rollouts say nothing about selection. Stage 3 runs the agent in the simulator and updates the selection policy from the outcome of complete routes. In this stage the privileged expert answers the agent's questions, so the return of a rollout reflects which questions were asked, not how well they were answered. **At evaluation no expert answers are available and the model answers its own questions; every number we report is obtained this way.**

Per-step reward. The reward of frame t scores the decision and penalises infractions. The decision consists of two control keys, direction and speed, written $\hat{u}_{t,j}$ for $j \in \{1, 2\}$; the expert's keys are $u^{\star}_{t,j}$. With $p_t \leq 0$ the infraction penalty of frame t (defined below),

$$r_t = \begin{cases} r_{\text{fmt}} + p_t, & \text{if the output cannot be parsed,} \\ \sum_j \text{score}\big(\hat{u}_{t,j},\, u^{\star}_{t,j}\big) + p_t, & \text{otherwise,} \end{cases} \tag{3}$$

where r_{fmt} is a fixed penalty and score is a lookup table that rewards agreeing with the expert and penalises over-driving more than under-driving (Appendix A.2).

The infraction penalty is not hand-weighted. The benchmark computes the Driving Score as route completion RC (the fraction of the route driven) times an official coefficient $\gamma_k \in (0, 1)$ for every infraction of type k: $\text{DS} = \text{RC} \cdot \prod_k \gamma_k^{n_k}$, where n_k counts the infractions of type k. Taking the logarithm turns the product into a sum with one term per infraction, and we charge each frame its own terms:

$$\ln \text{DS} \;=\; \ln \text{RC} \;+\; \sum_k n_k \ln \gamma_k, \qquad p_t \;=\; \sum_{k \in E_t} w_k, \quad w_k = \kappa \ln \gamma_k, \tag{4}$$

where E_t lists the infractions at frame t and the constant κ puts the penalties on the scale of the key scores. The relative weight of each infraction type thus comes from the benchmark, not from us, and the same construction works for any score that is a product of penalty coefficients.

One more problem is timing. An infraction is recorded at a single frame, e.g. a collision at the moment of impact, but the decision that caused it was taken about a braking distance earlier. As we have no value function to carry the penalty backwards, we copy each point-in-time penalty to the W frames before it, with a weight that falls linearly with the distance d:

$$\tilde{r}_t \;=\; r_t \;+\; \lambda \sum_{d=1}^{W} \frac{W - d}{W} \sum_{k \in E_{t+d} \cap\, \mathcal{E}_{\text{pt}}} w_k, \tag{5}$$

where $\mathcal{E}_{\text{pt}}$ is the set of point-in-time infraction types (collisions and traffic violations), W covers about one braking distance, and λ scales the copies. The impact frame keeps its full penalty. Episode-level failures (a scenario timeout, a blocked vehicle, a route deviation) have no single responsible frame and are not spread back.

Hybrid advantage. Training must combine two signals. The Driving Score is what we care about, but it arrives once per route and depends on a few decisive moments; alone, it gives almost no learning signal to most selections. The per-step reward $\tilde{r}_t$ is dense and immediate, but it judges each frame in isolation; alone, it reduces the policy to frame-by-frame imitation of the expert, which is exactly the short-sighted objective of §3.1. We therefore add the two. For frame t of rollout i,

$$A_t \;=\; \frac{\tilde{r}_t - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}} + \varepsilon} \;+\; \alpha\, \frac{\text{DS}_i - \mu_G}{\sigma_G + \varepsilon}, \tag{6}$$

where $\mu_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}$ are the mean and standard deviation of $\tilde{r}_t$ over all frames of the training batch, DS_i is the Driving Score of rollout i, μ_G and σ_G are the mean and standard deviation over the G rollouts of the same route, α weights the second term, and ε avoids division by zero. The second term compares a rollout only with other rollouts of the same route, so it rewards asking questions in a way that drives this route better, not driving an easy route. We add the terms rather than multiply them: with a product, an episode whose score is near its group mean would give all of its frames an advantage near zero. We optimise with Group Relative Policy Optimization (GRPO; Shao et al., 2024b); our change is the advantage, since standard GRPO uses only the group-normalised final return. Optimiser details are in Appendix A.3.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate on Bench2Drive-VL (Jia et al., 2026), a closed-loop benchmark in CARLA (Dosovitskiy et al., 2017) whose privileged expert generates ground-truth answers online while the

Table 1: Closed-loop driving and reasoning on *dev10*, with camera input and the benchmark's default actor list throughout. *Chain*: the benchmark's fixed chain (*yes*), committing without intermediate questions (*no*), a trained reasoning policy (*learned*). *Reasoning*: average over the benchmark's eight scored question types (Table A5). † the benchmark's published numbers; DriveCommenter is its privileged expert, an upper bound excluded from the comparison; ‡ our own reproductions. Best per column in bold; ours shaded.

		Driving				Reasoning
Model	Chain	DS ↑	SR (%) ↑	Effic. ↑	Comf. ↑	Avg ↑
DriveCommenter†	–	100.00	100.0	154.91	12.48	100.00
Qwen2.5-VL-3B†	yes	42.24	0.0	65.61	36.20	52.85
Qwen2.5-VL-3B†	no	58.93	20.0	74.72	37.67	50.05
Gemma3-4b-it†	yes	15.04	0.0	60.10	34.79	24.07
Gemma3-4b-it†	no	37.53	0.0	120.45	32.03	40.39
InternVL3-2B†	yes	33.00	20.0	48.36	65.48	54.43
InternVL3-2B†	no	29.92	0.0	51.18	**77.45**	52.14
AlphaDrive (Jiang et al., 2025)‡	learned	66.68	30.0	110.31	33.24	62.34
DriveAgent-R1 (Zheng et al., 2026a)‡	learned	59.79	30.0	113.97	33.06	57.99
DynaChain-RL (ours)	learned	**79.15**	**50.0**	**137.37**	21.60	**62.40**

agent drives, and which scores driving and reasoning on the same episodes. The supervised stages use its own annotations (§3.3); closed-loop reinforcement learning runs on a training pool of 116 routes, and all results are reported on its ten-scenario route set (*dev10*), which its own published numbers also use. Dataset construction and sizes are given in Appendix A.4.

Evaluation metrics. Driving Score is the primary metric: route completion multiplied by a penalty coefficient for each infraction, as decomposed in §3.3. We also report the Success Rate, Efficiency and Comfortness the benchmark defines. Reasoning quality is scored per question on a 0–100 scale by the benchmark's own protocol against the expert's answers, with Qwen3-235B (Yang et al., 2025) as an LLM judge (Zheng et al., 2023) where the comparison is not exact; we report its eight scored question types and their average. Further evaluation details are given in Appendix A.5.

Implementation details. All ablation variants share the same base model, question catalogue and frozen controller, and differ only in how questions are selected and how that selection is trained. The base model is Qwen2.5-VL-3B-Instruct (Bai et al., 2025b), with the vision tower frozen and only low-rank adapters (Hu et al., 2022) trained. Stage three optimises with GRPO at a learning rate of 2×10^{-5}, five rollouts per route and three routes per step, with α=0.2 in equation 6 and point-in-time penalties spread back over a window of eight preceding frames. Remaining hyperparameters and the training set-up are given in Appendix A.3 and Appendix A.4.

4.2 Main Results

The full chain at every frame drives substantially worse than asking nothing: 42.24 against 58.93 (Table 1). Trained against closed-loop Driving Score, DynaChain-RL reaches 79.15 — 20.22 points above the no-chain configuration, and 12.47 above the stronger of the two reproduced adaptive-reasoning methods (66.68 for AlphaDrive, 59.79 for DriveAgent-R1) — while asking 2.52 questions per decision against the fixed chain's seven, and without abandoning structured reasoning.

The improvement does not come from answering better: Table 1 shows twice that reasoning quality does not track driving quality. In the benchmark's own numbers, the fixed chain answers its questions *more* accurately on average than the no-chain configuration (52.85 against 50.05) and nevertheless drives 16.69 points worse. AlphaDrive attains an average of 62.34, within 0.06 of our 62.40, while its Driving Score is 12.47 points lower. The first is impossible if Driving Score is a monotone function of answer quality, the assumption §3.1 identifies as the point of failure. The gain from closed-loop training therefore shows not in the reasoning score but in how the agent acts: which questions it raises, and what it commits to once they are answered.

Table 2: Ablations. *Sel.*: questions chosen by the trained policy; *CL*: rollouts executed in the simulator rather than open-loop; *Dense* / *Epis.*: the two terms of equation 6. The base-model row (†) is the benchmark's published no-chain configuration. Best per column in bold; ours shaded.

	Training			Reward		Driving				Reasoning
Variant	Sel.	RL	CL	Dense	Epis.	DS ↑	SR (%) ↑	Effic. ↑	Comf. ↑	Avg ↑
Base model, no chain†	×	×	×	×	×	58.93	20.0	74.72	**37.67**	50.05
SFT only (DynaChain-SFT)	✓	×	×	×	×	60.44	20.0	122.83	20.22	60.98
Open-loop RL	✓	✓	×	✓	×	68.99	40.0	129.34	24.14	63.44
Closed-loop RL, episodic	✓	✓	✓	×	✓	49.75	0.0	**145.46**	21.85	60.15
Closed-loop RL, dense	✓	✓	✓	✓	×	64.15	30.0	122.91	20.04	**64.18**
Ours, random selection	×	✓	✓	✓	✓	40.74	0.0	93.35	26.13	61.56
DynaChain-RL (ours)	✓	✓	✓	✓	✓	**79.15**	**50.0**	137.37	21.60	62.40

4.3 Ablations

Learned selection, not sparsity, carries the result. Replacing the trained policy's choices with a random subset of the same average size leaves 40.74, below the base model, although every trained component stays in place and only *which* questions are asked has changed (Table 2). Its reasoning average (61.56) is within one point of ours, so the 38.41-point gap in Driving Score is attributable to the selection. Nor is what is learned a shorter fixed chain: over the 700 decision steps of *dev10* the policy uses 54 distinct tool sets and changes the set at half of all steps.

Supervision moves reasoning; closed-loop training moves driving. The supervised stages raise the reasoning average 10.93 points above the base model while Driving Score moves by 1.51. Closed-

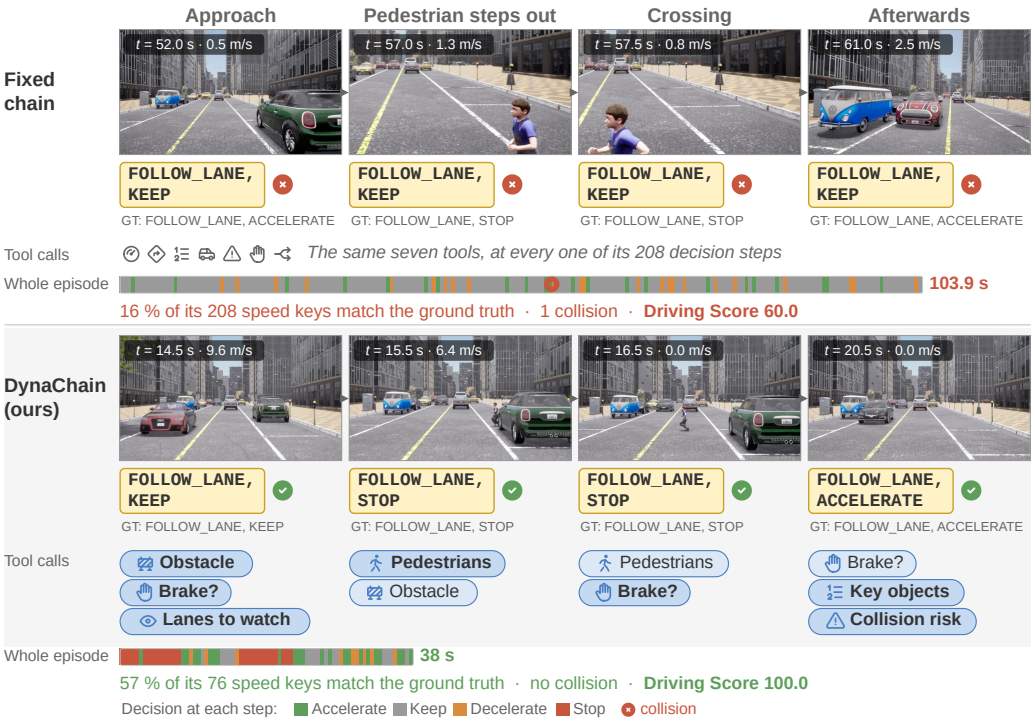

Figure 4: The scenario of Figure 1 over two whole episodes, aligned by phase. Under each frame the decision is given as its two control keys and checked against the ground-truth (GT) keys, with the tools DynaChain called there (newly selected ones in bold). The strip closing each row is the whole episode, one segment per decision step on a scale shared by both rows. Icons as in Figure 2.

loop reinforcement learning then adds 18.71 points of Driving Score while the reasoning average rises by only 1.42. The decoupling of §4.2 therefore also holds within a single training pipeline.

Both reward terms are required, and training must be closed-loop. Without the dense term the policy reaches 49.75, below supervision alone: the route-level signal is too sparse to shape individual selections. Without the episodic term it reaches 64.15, 15.00 short, because it then learns to agree with the expert frame by frame rather than to drive the route well. Keeping the dense term but training without the simulator — open-loop RL — reaches 68.99, 10.16 below ours.

4.4 Qualitative Analysis

Figure 4 follows one scenario, a pedestrian stepping out from behind a parked car, under the fixed chain and under DynaChain. First, the two call different tools. The fixed chain calls its seven question tools at every one of its 208 steps; at the step where the pedestrian enters the lane, the answers rank the pedestrian last among the important objects and report a speed limit that does not exist (Figure 1). DynaChain calls 2.8 tools per step here and adapts them to the situation, calling the pedestrian tool at seven steps only — exactly the seven at which the ground truth asks the car to stop for the pedestrian. Second, they decide differently. The fixed chain decides `KEEP` at 78% of its steps and `STOP` at none, agreeing with the ground-truth speed key at 16% of steps; it crawls through the route at about 1 m/s, collides with another vehicle after 56 s, and needs 104 s to finish 117 m. DynaChain matches that key at 57% of steps, stops while the pedestrian crosses, resumes once the road is clear, and completes the route in 38 s with no infraction of any kind.

5 Conclusion

We have addressed a failure of fixed reasoning chains in closed-loop driving: asked identically at every frame, such a chain drives worse than asking nothing at all. Formalising reasoning-chain selection as a closed-loop Markov decision process shows why an objective measured where the choice is made optimises a different quantity from the one that matters. On that basis we designed a closed-loop agentic reinforcement-learning procedure that trains the selection policy against the driving outcome, extending GRPO with a hybrid advantage whose dense term takes its infraction weights from the benchmark's own score. DynaChain raises Driving Score from 58.93 to 79.15 while asking 2.52 questions per decision rather than seven. We expect the formulation to transfer to other agentic settings in which the value of a reasoning step is realised only in what follows it.

AI use statement

In this work, we used generative AI tools to generate synthetic data, to assist in executing the research and to aid in writing. The stage-two selection traces are synthetic: they are proposed by an LLM teacher and then verified against the ground truth and inspected by the authors (Appendix A.4). Generative AI tools helped implement the training, evaluation and data-processing code, and gave feedback on the design of experiments and the interpretation of results. They were also used to polish the writing of the paper, help prepare figures and format references. We have reviewed all AI-assisted work: code was tested, every reported number was traced to the raw evaluation logs, every citation was checked against the published version of the cited paper, and all text and figures were revised by the authors. Separately, large language models are components of the method and its evaluation — the stage-two teacher and the benchmark's judge — as described in §4 and Appendix A.4.

Reproducibility statement

Anonymised source code for all three training stages, the closed-loop evaluation and the scripts that produce every table is provided as supplementary material, with setup instructions; the third-party components it builds on (CARLA, the benchmark and veRL) are public and not redistributed. All models used are open-weight. The question catalogue and all prompts are given verbatim in Appendix A.1, the reward in Appendix A.2, the objective and hyperparameters in Appendix A.3, the construction of the training data and route pool in Appendix A.4, and the evaluation protocol in Appendix A.5.

REFERENCES

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

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025b.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do NOT think that much for 2+3=? on the overthinking of long reasoning models. In *International Conference on Machine Learning (ICML)*, 2025.

Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. TransFuser: Imitation with transformer-based sensor fusion for autonomous driving. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11):12878–12895, 2023.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Conference on Robot Learning (CoRL)*, 2017.

Gongfan Fang, Xinyin Ma, and Xinchao Wang. Thinkless: LLM learns when to think. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

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 *IEEE/CVF International Conference on Computer Vision (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 *European Conference on Computer Vision (ECCV)*, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638, 2025.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, Lewei Lu, Xiaosong Jia, Qiang Liu, Jifeng Dai, Yu Qiao, and Hongyang Li. Planning-oriented autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Jyh-Jing Hwang, Runsheng Xu, Hubert Lin, Wei-Chih Hung, Jingwei Ji, Kristy Choi, Di Huang, Tong He, Paul Covington, Benjamin Sapp, Yin Zhou, James Guo, Dragomir Anguelov, and Mingxing Tan. EMMA: End-to-end multimodal model for autonomous driving. *Transactions on Machine Learning Research*, 2025.

Ayesha Ishaq, Jean Lahoud, Ketan More, Omkar Thawakar, Ritesh Thawkar, Dinura Dissanayake, Noor Ahsan, Yuhao Li, Fahad Shahbaz Khan, Hisham Cholakkal, Ivan Laptev, Rao Muhammad Anwer, and Salman Khan. DriveLMM-o1: A step-by-step reasoning dataset and large multimodal model for driving scenario understanding. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025.

Xiaosong Jia, Zhenjie Yang, Qifeng Li, Zhiyuan Zhang, and Junchi Yan. Bench2Drive: Towards multi-ability benchmarking of closed-loop end-to-end autonomous driving. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024.

Xiaosong Jia, Yuqian Shao, Zhenjie Yang, Qifeng Li, Zhiyuan Zhang, and Junchi Yan. Bench2Drive-VL: Benchmarks for closed-loop autonomous driving with vision-language models. *arXiv preprint arXiv:2604.01259*, 2026.

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

Bo Jiang, Shaoyu Chen, Qian Zhang, Wenyu Liu, and Xinggang Wang. AlphaDrive: Unleashing the power of VLMs in autonomous driving via reinforcement learning and reasoning. *arXiv preprint arXiv:2503.07608*, 2025.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O. Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Conference on Language Modeling (COLM)*, 2025.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. In *International Conference on Learning Representations (ICLR)*, 2023.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2023.

Qifeng Li, Xiaosong Jia, Shaobo Wang, and Junchi Yan. Think2Drive: Efficient reinforcement learning by thinking with latent world model for autonomous driving (in CARLA-V2). In *European Conference on Computer Vision (ECCV)*, 2024.

Chenwei Lou, Zewei Sun, Xinnian Liang, Meng Qu, Wei Shen, Wenqi Wang, Yuntao Li, Qingping Yang, and Shuangzhi Wu. AdaCoT: Pareto-optimal adaptive chain-of-thought triggering via reinforcement learning. *arXiv preprint arXiv:2505.11896*, 2025.

Ana-Maria Marcu, Long Chen, Jan Hünermann, Alice Karnsund, Benoit Hanotte, Prajwal Chidananda, Saurabh Nair, Vijay Badrinarayanan, Alex Kendall, Jamie Shotton, Elahe Arani, and Oleg Sinavski. LingoQA: Visual question answering for autonomous driving. In *European Conference on Computer Vision (ECCV)*, 2024.

Ming Nie, Renyuan Peng, Chunwei Wang, Xinyue Cai, Jianhua Han, Hang Xu, and Li Zhang. Reason2Drive: Towards interpretable and chain-based reasoning for autonomous driving. In *European Conference on Computer Vision (ECCV)*, 2024.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, Jie Tang, and Yuxiao Dong. WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2025.

Kangan Qian, Sicong Jiang, Yang Zhong, Ziang Luo, Zilin Huang, Tianze Zhu, Kun Jiang, Mengmeng Yang, Zheng Fu, Jinyu Miao, Yining Shi, He Zhe Lim, Li Liu, Tianbao Zhou, Hongyi Wang, Huang Yu, Yifei Hu, Guang Li, Guang Chen, Hao Ye, Lijun Sun, and Diange Yang. AgentThink: A unified framework for tool-augmented chain-of-thought reasoning in vision-language models for autonomous driving. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025.

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

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Hao Shao, Yuxuan Hu, Letian Wang, Guanglu Song, Steven L. Waslander, Yu Liu, and Hongsheng Li. LMDrive: Closed-loop end-to-end driving with large language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024a.

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. In *European Conference on Computer Systems (EuroSys)*, 2025.

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*, 2026.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Chonghao Sima, Katrin Renz, Kashyap Chitta, Li Chen, Hanxue Zhang, Chengen Xie, Jens Beißwenger, Ping Luo, Andreas Geiger, and Hongyang Li. DriveLM: Driving with graph visual question answering. In *European Conference on Computer Vision (ECCV)*, 2024.

Alex Su, Haozhe Wang, Weiming Ren, Fangzhen Lin, and Wenhu Chen. Pixel Reasoner: Incentivizing pixel space reasoning via curiosity-driven reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Xiaoyu Tian, Junru Gu, Bailin Li, Yicheng Liu, Yang Wang, Zhiyong Zhao, Kun Zhan, Peng Jia, Xianpeng Lang, and Hang Zhao. DriveVLM: The convergence of autonomous driving and large vision-language models. In *Conference on Robot Learning (CoRL)*, 2024.

Shihao Wang, Zhiding Yu, Xiaohui Jiang, Shiyi Lan, Min Shi, Nadine Chang, Jan Kautz, Ying Li, and Jose M. Alvarez. OmniDrive: A holistic vision-language dataset for autonomous driving with counterfactual reasoning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025a.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025b.

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

Zhenhua Xu, Yujia Zhang, Enze Xie, Zhen Zhao, Yong Guo, Kwan-Yee K. Wong, Zhenguo Li, and Hengshuang Zhao. DriveGPT4: Interpretable end-to-end autonomous driving via large language model. *IEEE Robotics and Automation Letters*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui

Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023.

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 *International Conference on Learning Representations (ICLR)*, 2026a.

Ziwei Zheng, Minghao Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and Xing Yu. DeepEyes: Incentivizing "Thinking with Images" via reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2026b.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. ArCHer: Training language model agents via hierarchical multi-turn RL. In *International Conference on Machine Learning (ICML)*, 2024.

Zewei Zhou, Tianhui Cai, Seth Z. Zhao, Yun Zhang, Zhiyu Huang, Bolei Zhou, and Jiaqi Ma. AutoVLA: A vision-language-action model for end-to-end autonomous driving with adaptive reasoning and reinforcement fine-tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

A Appendix

This appendix follows the order of the main text. §A.1 and §A.2–§A.3 complete the description of the method in §3: the question catalogue and the three prompts, the reward table, and the optimiser. §A.4 and §A.5 give the training set-up and the evaluation protocol behind §4. §A.6 and §A.7 report results that did not fit the main text: the per-question reasoning scores, and the statistics of what the learned policy actually asks.

A.1 Question Catalogue and Prompts

The catalogue. The agent may ask any of the thirteen questions of Table A1. They are a subset of the benchmark's fifty annotated question types, chosen so that each one is answerable from the current frame alone and can change a control decision; the benchmark's remaining questions either restate another question at a different granularity, address a named actor that the agent has not yet identified, or ask for a trajectory rather than a fact. Two further questions are not selectable, because they constitute the commitment itself: question 43 (*what is the correct action for the ego vehicle to take now?*) and question 50, which asks for the two control keys directly. The fixed chain of the benchmark calls seven of the thirteen at every frame (questions 19, 15, 24, 7, 13, 47 and 8) and then answers 43 and 50; our agent selects any subset of the thirteen, in any order, and answers the same two.

The three prompts. A decision step uses the three prompts of Prompts A.1–A.3. They are byte-identical across the stage-two supervision, the closed-loop reinforcement learning and the evaluation harness; any drift between them would train the policy on a different input distribution than it is tested on, so the implementation asserts their identity. Line breaks reflow to the page width; the exact strings are in the supplementary code.

The *selection prompt* (Prompt A.1) is the system message of the policy loop. It is followed by a user message carrying the two concatenated camera images and the ego context, and is re-used unchanged for every turn of the same frame.

Prompt A.1: Selection prompt (system message)

You are the driving policy of an autonomous vehicle. Every control step you must commit to a Direction Key and a Speed Key for the car to execute right now.

Before committing you MAY consult sub-questions ("tools"); each is answered with one short fact added to your notes. Consulting costs time: consult a tool ONLY when, without its answer, you might commit to a different Direction or Speed Key. Do not consult tools that merely confirm what you already know or give generic advice. Commit as soon as your notes fix both keys.

Each step, reason in `<think>...</think>` about what you see, what is still genuinely uncertain for THIS decision, and what a tool's answer would change. Then output exactly ONE of:

`<tool>NAME</tool>` — consult this tool (NAME must be from the menu)
`<answer>Direction Key = <DIR>, Speed Key = <SPEED></answer>` — commit now, nothing after

`DIR` : `FOLLOW_LANE` | `CHANGE_LANE_LEFT` | `CHANGE_LANE_RIGHT` (following a road)
`GO_STRAIGHT` | `TURN_LEFT` | `TURN_RIGHT` (in a junction)
`DEVIATE_LEFT` | `DEVIATE_RIGHT` (small shift within the lane)
`SPEED`: `KEEP` | `ACCELERATE` | `DECELERATE` | `STOP`

Available tools:
the thirteen names of Table A1, each with a one-line description

The user message of the first turn supplies the observation (Prompt A.2). `Ego state` and `Navigation` come from the simulator, `Nearby actors` from the benchmark's textual perception summary, which the benchmark builds from the simulator's annotations and supplies to every method in its default configuration, and `Special` appears only when the benchmark flags a scenario state.

Prompt A.2: Observation (user message, first turn)

The two concatenated images below are from all cameras attached to the ego vehicle on current frame.

[front-camera composite] [rear-camera composite]

`Ego state: speed {speed} m/s.`
`Navigation: {command}.`
`Special: {state}.` (only when flagged)
`Nearby actors:`
[one line per actor: class, position, distance, speed]

What do you do?

When the policy names a tool, that question is answered by a *separate* call (Prompt A.3) that sees the images and the scene facts but not the policy's reasoning, so an answer cannot be shaped by the decision it is meant to inform. Its reply is then appended to the conversation as one line, and the loop continues.

Prompt A.3: Isolated tool call (single-turn, independent conversation)

The two concatenated images below are from all cameras attached to the ego vehicle on current frame.

[front-camera composite] [rear-camera composite]

Use information above to answer:
`Basic infos of the actors: {actor_json}`
[question-specific condition line, from the benchmark's `generate_condition`*]*
[the question text of Table A1]

The loop runs for at most K=7 turns, the committing turn included, so at most six questions are asked. A turn that is neither a valid `<tool>` nor a valid `<answer>`, and a repeated tool, are both rejected with a one-line correction; such a turn asks nothing, but it uses up a turn and sets the format flag that the reward of §3.3 gates on.

A.2 Reward Design

This appendix gives the numerical values behind the per-step reward of §3.3. All of them are fixed before training and never tuned against a result.

Format gate. A turn sequence counts as malformed if the policy never emits a parsable `<answer>` within its K turns, or emits a tool name outside the catalogue, or repeats one. Such a frame receives $r_{\text{fmt}} = -6.0$ in place of the key terms. The value is chosen to lie below the worst legal pair of key scores ($-5.0 - 0.5 = -5.5$) so that emitting nothing is never preferable to committing badly, which would otherwise make the format gate an escape hatch from a dangerous frame.

Control keys. The speed key is scored by the signed distance between the emitted and the expert's level on the ordered scale `STOP` < `DECELERATE` < `KEEP` < `ACCELERATE`, with the asymmetry of Table A2: over-driving is penalised roughly 2.5 times as heavily as under-driving by the same number of levels. The direction key is scored +0.5 when it matches the expert and −0.5 otherwise, without an ordering, since its values are not comparable on a scale.

The asymmetry is set so that the decision threshold it induces is defensible rather than convenient. Ignoring the direction term, a policy that believes the expert's key is `STOP` with probability p prefers to stop when $p(+1.0)+(1-p)(-2.0) > p(-5.0)+(1-p)(+1.0)$, that is when $p > 1/3$: it stops when stopping is a one-in-three chance of being right. A milder conservative side makes that threshold implausibly low, and also makes freezing nearly free, because the direction term is correct on almost every frozen frame and cancels most of the speed penalty.

Table A1: The thirteen selectable questions, with the benchmark's own question identifier and the name by which the policy refers to them. Question text is the benchmark's, abbreviated where it interpolates the scene; questions 43 and 50 are the commitment and are not selectable.

ID	Name	Question
6	`traffic_light`	What is the state of the traffic light?
7	`speed_limit`	What is the current speed limit?
15	`signs_and_actions`	Identify all traffic lights and signs affecting the ego vehicle. Based on these, what actions should the ego vehicle take respectively?
19	`important_objects`	What are the important objects in the scene? List them from most to least important.
24	`vehicles_motion`	The important vehicles are . . . What is the rough moving speed and moving direction of them?
1	`vru_check`	How many pedestrians are there?
47	`collision_risk`	The important vehicles are . . . List potential overlap vehicles, overlap reasons and the actions that could lead to a collision.
11	`obstacle_on_road`	Is there an obstacle on the current road?
41	`env_hazards`	Apart from vehicles, pedestrians and the weather, what other factors could pose hazards, and what should the ego vehicle do about them?
8	`need_brake`	Does the ego vehicle need to brake? Why?
13	`must_change_lane`	Must the ego vehicle change lane or deviate from the lane now? Why?
31	`lane_change_feasibility`	In which direction is the ego car allowed to change lanes?
36	`lanes_to_watch`	The ego vehicle wants to . . . Which lanes must it watch?

Table A2: Speed-key score as a function of err = level($\hat{u}$) − level($u^\star$). Negative values of err are too conservative, positive too aggressive.

err	−3	−2	−1	0	+1	+2	+3
score	−2.0	−1.2	−0.5	**+1.0**	−1.2	−2.8	−5.0
	← too conservative			correct	too aggressive →		

Infraction weights. Table A3 lists $w_k = \kappa \ln \gamma_k$ for every infraction the benchmark scores, with $\kappa = 12$. Because the weights are a fixed transform of the benchmark's own Driving Score coefficients γ_k, their relative severity is inherited from the metric rather than chosen by us; κ sets only the scale against the key terms.

Two infractions, a blocked vehicle and a route deviation, have no coefficient, because they do not scale the score but *end the route*, forfeiting the completion still ahead, which costs more than any single collision. The two are therefore assigned the most severe weight in the table, which is a conservative floor rather than an estimate. A scenario timeout, by contrast, has an official coefficient and is weighted by it. Driving outside the route lanes reduces route completion rather than the score; it has no coefficient either and is kept as a mild hand-set shaping term.

One infraction is deliberately left unscored. The benchmark itself marks minimum-speed violations as unused in its penalty table, so they do not enter the Driving Score, and they are not a usable signal of driving too slowly either: in the validation rollouts taken before the first update they fired on 95% of all recorded infraction events, including eighteen times on a route that nevertheless finished with a perfect Driving Score of 100. A counter that fires on a flawless episode cannot distinguish cautious driving from bad driving, so rewarding it would add noise rather than pressure to keep up with traffic.

Table A3: Infraction weights. γ_k is the benchmark's Driving Score coefficient and $w_k = \kappa \ln \gamma_k$ with $\kappa = 12$. The last two end the route instead of scaling the score and are assigned the table's most severe weight; the lane term is hand-set. *Spread back* marks the infractions attributed to a single frame, whose weight is additionally carried back over a window of eight preceding frames as described below; the rest describe a state rather than a moment and are charged only where they occur.

Infraction	γ_k	w_k	Spread back
Collision with a pedestrian	0.50	−8.3	✓
Collision with a vehicle	0.60	−6.1	✓
Collision with the layout	0.65	−5.2	✓
Red-light violation	0.70	−4.3	✓
Failure to yield to an emergency vehicle	0.70	−4.3	✓
Stop-sign violation	0.80	−2.7	✓
Scenario timeout	0.70	−4.3	
Driving outside the route lanes	—	−1.5	
Vehicle blocked	—	−8.3	
Route deviation	—	−8.3	

Spreading penalties back. A CARLA infraction is attributed to the single frame of impact, so in an episode almost every frame carries $p_t = 0$ and the penalty is a lone spike. The decision that caused it — continuing to drive at the obstacle — was made about a braking distance earlier, and with no value bootstrap a spike at impact leaves the approach frames with almost no gradient. Each point-in-time infraction therefore adds a decayed copy of its weight to the frames in a window of W=8 before it, linearly decayed as $(W - d)/W$ at offset d (so the copy at d=W is zero) and scaled by λ=0.5; the impact frame keeps its full weight. The copies add $1.75\, w_k$, so one infraction is charged $2.75\, w_k$ in total. Episode-level failures are a state rather than a moment and are not spread back, and driving outside the route lanes already fires on every off-lane frame.

Spreading penalties back changes behaviour, not only credit. Before it was added, the policy drove through static obstacles instead of braking for them, scoring 8.4 on *dev10*'s parking-exit route where the policy we report scores 100: an isolated penalty at the frame of impact is too distant from the decision that caused it to carry gradient back to it.

Magnitudes. Per-frame values that are individually reasonable can still make a degenerate policy profitable over an episode. We therefore verify the totals for the two such policies available here: stopping permanently, which incurs no infraction, and emitting nothing, which commits no wrong key. Over a forty-frame episode, driving correctly throughout yields about +60, and the same episode with one vehicle collision about +43. Stopping until the scenario times out at eighty seconds, roughly 160 frames, yields about −244, and unparsable output at every frame about −240. Both degenerate policies are thus penalised by an order of magnitude more than a collision costs, and neither is competitive with driving.

A.3 Optimisation

Objective. Training maximises the clipped surrogate of Group Relative Policy Optimization (Shao et al., 2024b) with the advantage of equation 6 substituted for the group-normalised terminal return. For a group of G rollouts on the same route, let y_i be the tokens generated in rollout i, $y_{i,\tau}$ its τ-th token and $s_{i,\tau}$ the context before it. We maximise

$$\mathcal{J}(\theta) \;=\; \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{\tau=1}^{|y_i|}\min\Big(\rho_{i,\tau}A_{i,\tau},\; \mathrm{clip}(\rho_{i,\tau},1-\epsilon_{\mathrm{c}},1+\epsilon_{\mathrm{c}})\,A_{i,\tau}\Big)\right] \;-\; \beta\,\mathbb{D}_{\mathrm{KL}}[\pi_\theta \,\|\, \pi_{\mathrm{ref}}]\,, \tag{7}$$

where $\rho_{i,\tau} = \pi_\theta(y_{i,\tau} \mid s_{i,\tau})/\pi_{\theta_{\mathrm{old}}}(y_{i,\tau} \mid s_{i,\tau})$ compares the current policy with the policy $\pi_{\theta_{\mathrm{old}}}$ that collected the rollouts, $A_{i,\tau}$ is the advantage equation 6 of the frame that token τ belongs to, ϵ_{c} is the clipping range, and β weights a KL penalty that keeps π_θ close to the reference policy π_{ref}, the stage-two model. The loss is taken over response tokens only: an assistant turn is a selection

or a commitment the policy produced, whereas the observation, the tool replies and the correction messages are environment text and are masked out.

Two properties of equation 6 bear stating. The dense term is normalised over the whole batch rather than within a group, because a frame's reward is comparable across routes — it grades a control decision against the expert — whereas the Driving Score is not, which is why the episodic term is group-relative. And the Kullback–Leibler term enters the loss rather than the reward, so it does not interact with the reward magnitudes of Appendix A.2.

Hyperparameters. Table A4 lists the settings of the reported run. Only low-rank adapters are trained; the vision tower is excluded from adaptation entirely, so the visual representation is the supervised model's throughout.

Table A4: Stage-three hyperparameters. Training runs on veRL (Sheng et al., 2025); rollout is by vLLM (Kwon et al., 2023) inside the CARLA loop, one route per rollout.

Setting	Value
Algorithm	GRPO with the advantage of equation 6
Episodic weight α	0.2
Rollouts per route G	5
Routes per optimisation step	3
Optimisation steps trained	60 ($\approx$ 1.5 epochs over the pool)
Reported checkpoint	step 50 ($\approx$ 1.3 epochs)
Learning rate	2×10^{-5}
Mini-batch size	32 frames
KL coefficient β	10^{-3}, low-variance estimator
KL placement	loss, not reward
Adapter rank / scaling	32 / 64
Adapted modules	all attention and MLP projections
Excluded from adaptation	the vision tower
Precision / sharding	bf16, FSDP2, gradient checkpointing
Rollout temperature	0.7
Max prompt / response length	8192 / 4096 tokens
Max turns per frame K	7

The rollout temperature is not left at a default, because the group-relative term of equation 6 only means anything if the G rollouts of a group actually differ. At temperature zero they differ only by simulator noise, the group's Driving Scores are then effectively the same, and the term is zero for every frame. Raise the temperature too far and the opposite failure appears: the commitment format learned in stage two breaks down, and the format gate fires instead of the key scores.

We therefore chose it offline, on held-out frames and without running the simulator. At each candidate temperature we resampled the same frame repeatedly and measured two things: how often the output still parsed, and whether the resamples actually disagreed — different committed keys, or different sets of questions asked. We took the highest temperature at which the output still parsed on more than 95% of samples while the committed keys genuinely varied across resamples, which is 0.7.

A.4 Training Set-up and Data

All three stages train low-rank adapters on Qwen2.5-VL-3B-Instruct with the vision tower frozen; stage three is in Table A4. The two supervised stages share their optimisation entirely: both are single-epoch LoRA runs of rank 16, scaling 32 and dropout 0.05, at a learning rate of 1×10^{-4} with cosine decay and 3% warmup, in bf16 on four GPUs with a global batch of 64 sequences (4 per device $\times$ 4 accumulation steps $\times$ 4 GPUs) and a budget of 589,824 pixels per image. They differ in three things only: stage one trains 84,726 samples from the base model within a 4096-token limit, stage two 53,944 samples from the merged stage-one checkpoint within 5120.

Stage one: open-loop question answering. The corpus is produced by the benchmark's own annotation generator, which emits ground-truth answers to its fifty question types for every frame of the public driving dataset. Each sample is one question: a prompt carrying two concatenated camera composites — the three forward cameras and the three rearward ones — and the expert's answer as the target. Questions are independent here, with no chain context, matching the benchmark's own open-loop evaluation.

We split by scenario, not by frame, so that no frame of an evaluation scenario appears in training in any form. Of the 254 processable scenarios, 42 are held out and 212 are used for training. Frames are subsampled at a stride of three, since consecutive frames at 10 Hz are near-duplicates, giving 84,726 training samples and 9,219 held-out samples.

Stage two: distilled selection traces. Stage two teaches the format and a reasonable prior over which questions to ask; it does not teach the closed-loop behaviour, which is what stage three supplies.

We sample 4,671 frames from the stage-one training scenarios, weighted by the expert's speed key so that the rare decisive frames are not swamped by the common one: `STOP` is weighted 3, `DECELERATE` 2, `ACCELERATE` 1.5 and `KEEP` 1, drawn without replacement by the Efraimidis–Spirakis scheme. For each frame a strong teacher, Qwen3-VL-235B-A22B (Bai et al., 2025a), sees the images, every ground-truth answer and the correct control keys, and returns the smallest ordered set of questions that justifies those keys, with one sentence of reason per step.

A teacher that already knows the answer is not a reliable judge of what was required to reach it, so every set is verified against the frame in three steps. *Sufficiency*: a referee, the same Qwen3-VL-235B-A22B, is given only the ground-truth answers of the selected questions and must recover the control keys; if it cannot, it is asked what is missing, one question is added, and the frame is marked weak if it still fails. *Redundancy*: any selected question whose ground-truth answer is negative, empty or generic is provisionally dropped, and the drop is kept if the referee's keys are unchanged or closer. *Fallback*: an empty set on a frame whose action is not `KEEP` is marked weak. A safety rule overrides the teacher in one direction only — if the ground-truth answer to a question about lights, signs, pedestrians or obstacles has content, that question must appear — because a teacher reasoning backwards from a known answer systematically omits the question that produced it.

What survives the three checks is then inspected by the authors before it is used: an exhaustive programmatic pass re-parses every trace, verifies that its tool names are legal, unique and dependency-ordered, and checks that no reasoning step leaks an answer the agent has not yet requested; a sample stratified over speed keys and trace lengths is then read manually, and the batch is regenerated if that reading turns up a systematic fault.

The surviving traces are assembled into 53,944 supervised samples: 13,956 multi-turn selection traces (4,652 distinct traces, each repeated three times so that selection is not swamped by answering) and 39,988 single-turn tool answers under the same template the agent uses at inference. Loss is taken on every assistant turn, so both the selection turns and the commitment are trained. The adapter is trained on the stage-one checkpoint and merged into it before stage three; reinforcement learning never starts from the bare base model.

Stage three: the training route pool. The benchmark's route file holds 44 scenario types with about five route instances each. We characterised one instance of every type by running it three times with the stage-two policy, which sorted the types into eight completed on all three runs, sixteen with mixed outcomes and twenty failed on all three. We retained all sixteen mixed types and eight of the failed ones, and none of the eight already completed: a route the policy always completes yields G rollouts with the same Driving Score, and the group-relative term of equation 6 is then identically zero, so the route contributes nothing to the gradient.

The resulting 24 types are then expanded to 116 routes by adding every remaining route instance of those same types. Expanding by type rather than sampling fresh routes keeps the wall-clock cost of a step predictable, since duration and difficulty follow the scenario type, while raising the diversity each gradient step sees; a step draws three routes regardless of pool size, so a larger pool costs nothing per step.

No *dev10* route enters the pool at any point. We use *dev10* unchanged because it is the benchmark's own published evaluation set, and the baselines we compare against are its authors' numbers on exactly these routes.

A.5 Evaluation Protocol

All reported numbers are measured on *dev10*, the benchmark's own ten-route evaluation set. Its ten routes are one instance each of ten scenario types, spanning unprotected and signalised left turns, sequential lane changes, yielding to an emergency vehicle, static cut-ins, blocked intersections, roadside and construction obstacles, a parking exit, and a pedestrian crossing in front of parked cars.

Driving metrics. All four are the benchmark's own, and follow the Bench2Drive protocol (Jia et al., 2024) it inherits. **Driving Score** is route completion scaled by a multiplicative penalty for each infraction, with the coefficients of Table A3. **Success Rate** is the fraction of routes completed with an empty infraction list. **Efficiency** is the ego's mean speed as a percentage of the surrounding traffic's, and **Comfortness** the fraction of trajectory segments meeting the benchmark's acceleration and jerk thresholds.

Reasoning score. Reasoning quality is scored per question against the privileged expert's answer for the same frame, on a 0–100 scale, over the eight question types the benchmark scores. Each type carries its own scoring rule, and the judge model — Qwen3-235B (the A22B-Instruct-2507 checkpoint), as in §4 — is used only for well-defined sub-tasks within it, such as extracting the objects an answer names or the control command it implies:

- *Important objects.* The expert's list is ordered — objects with a role in the scenario first, then dangerous ones, then the rest by distance — and each object carries a weight that is higher for role and dangerous objects and decays with rank. The answer is scored by the product of a weighted F_1 over the objects it names and the NDCG of the order it puts them in, so naming the right objects in the wrong order is penalised, and inventing objects costs a fraction of a real one.
- *Multi-object reasoning.* Questions that ask for a set of objects and then something about each are scored by a weighted F_1 over the set times the mean graded correctness of the per-object statements. The penalty for an extra object depends on the question: a harmless extra costs little, whereas an extra vehicle named as a collision risk costs more than a real one, because there the extra object *is* the reasoning error.
- *Action and control keys.* The judge parses the control command implied by the answer, and the score is an F_1 against the expert's keys multiplied by a speed penalty that reflects severity rather than distance: answering `ACCELERATE` where the expert says `STOP` is penalised hardest, while `ACCELERATE` in place of `KEEP` is treated as mildly favourable.
- *Remaining types.* These are graded by the judge under question-specific guidelines — overestimating a speed limit is penalised more than underestimating it, and a braking question accepts several answers when the car is already stopped.

We change none of this. The fixed-chain and no-chain rows are the benchmark authors' published numbers. The two adaptive-reasoning methods we compare against were developed on other benchmarks, so we reproduced both ourselves, following each paper's own training recipe on our data, and mark those rows to distinguish them from published results.

A.6 Per-Question Reasoning Scores

Table A5 breaks the reasoning average of Table 1 into the benchmark's eight scored question types.

A.7 Tool-Selection Behaviour

Table A6 reports what the learned policy actually asks over the 700 decision steps of the ten reported *dev10* episodes. Three things are visible in it.

The policy has not collapsed onto a shorter fixed chain. It uses 54 distinct question sets and changes the set between one step and the next at exactly half of all consecutive steps, while asking 2.52

Table A5: Reasoning quality on *dev10*: per-question scores against the privileged expert's answers, and their average, over the benchmark's eight scored question types (Important Objects, Traffic Sign, Speed Limit, Collide Object, Change Lane, Brake, Action Description, Action Keys). Rows and marks as in Table 1.

Model	Chain	Imp.Obj	T.Sign	S.Limit	Col.Obj	C.Lane	Brake	A.Desc	A.Keys	Avg
DriveCommenter†	–	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Qwen2.5-VL-3B†	yes	53.62	75.47	27.58	5.21	81.81	**62.94**	**69.69**	46.51	52.85
Qwen2.5-VL-3B†	no	50.37	78.76	24.64	4.82	75.04	57.67	58.07	51.03	50.05
Gemma3-4b-it†	yes	60.52	21.24	19.34	7.29	25.07	30.32	15.27	13.47	24.07
Gemma3-4b-it†	no	55.82	17.03	**100.00**	6.02	7.11	27.81	59.69	49.64	40.39
InternVL3-2B†	yes	52.96	71.16	69.08	2.89	84.52	59.08	66.56	29.22	54.43
InternVL3-2B†	no	55.54	72.41	46.84	3.03	83.97	58.36	61.35	35.59	52.14
AlphaDrive‡	learned	78.30	**91.40**	98.50	4.40	**89.30**	45.40	39.60	51.90	62.34
DriveAgent-R1‡	learned	66.00	79.30	89.90	8.70	87.80	51.40	31.70	49.10	57.99
DynaChain-RL (ours)	learned	**82.50**	89.40	98.50	**8.90**	82.00	52.40	27.50	**58.00**	**62.40**

questions per step against the fixed chain's seven. Nor has it collapsed onto a single length: sets of one, two, three, four and five questions occur at 16%, 30%, 43%, 9% and 2% of steps respectively.

The questions it favours are the ones closest to the decision it has to make. *Does the ego vehicle need to brake?* is asked at 57% of steps and *must the ego vehicle change lane?* at 45% — the two questions whose answers bear directly on the speed and direction keys — while questions that mostly restate the scene are asked less often, and some almost never.

The policy also learns not to ask. The open-ended environmental-hazard question is never asked at any of the 700 steps, and the ground truth explains why: over the 4,671 frames of the stage-two pool the expert answers it either that no serious hazard is present (28.5%) or that a wet or flooded surface makes the car harder to control and it should drive carefully (70.0%), neither of which can move a control key. Its answer is very nearly constant in this benchmark, so asking it spends a turn for no information about the decision at hand.

Table A6: What the learned policy asks, over the 700 decision steps of the ten reported *dev10* episodes. Each question is given as a percentage of the steps at which it is asked; a step may ask several.

	% of steps
`need_brake`	57.1
`must_change_lane`	45.3
`important_objects`	32.6
`collision_risk`	29.0
`obstacle_on_road`	24.9
`signs_and_actions`	24.6
`lanes_to_watch`	13.3
`vehicles_motion`	9.6
`traffic_light`	6.9
`vru_check`	6.4
`speed_limit`	1.9
`lane_change_feasibility`	0.7
`env_hazards`	0.0