SuperGlance: Driving at a Glance with a Vision-Language Model

Jieyuan Pei[1,2,3,*], **Xin Yan**[4,*], **Mingwei Xu**[5], **Haiyi Zhang**[3], **Siyu Zou**[2], **Zhangyi Hu**[2,3], **Zhuoxuan Li**[6], **Yuguang Yang**[7], **Rongjunchen Zhang**[2,†], **Cheng Chi**[6], **Hui Xiong**[3,✉], **Yan Wang**[1,✉]

[1]Institute for AI Industry Research (AIR), Tsinghua University [2]HiThink Research
[3]The Hong Kong University of Science and Technology (Guangzhou)
[4]Agency for Science, Technology and Research (A*STAR), Singapore
[5]University of Washington [6]Renmin University of China [7]Beihang University
*Equal contribution †Project leader ✉Corresponding authors

Autonomous driving is turning to vision-language-action models (VLAs). They answer and plan by decoding, one step at a time. Each step is another forward pass, so a plan can take over a second of GPU time. Embodied AI often pairs such a slow, reasoning System 2 with a fast System 1 that acts. Human drivers plan at a glance, with System 1. Faster designs usually put System 1 outside the vision-language model (VLM), in a planner without its abilities. Yet before a VLM writes its first token, it has already read the scene into one hidden state. We show that, once trained, this hidden state is enough to drive and to answer questions about what the model sees. SuperGlance reads it to turn a decoder-only VLM into an encoder-style driving policy. Its linear head scores a fixed set of 5,273 candidate trajectories. With the set fixed, rewards are computed offline, so supervised targets can be reward-weighted and reinforcement learning maximizes the exact expected reward. Before and during policy training, multiple-choice training keeps the VLM's visual abilities in one pass, so the policy still answers questions about driving scenes. Training questions are built against shortcuts. Each must be answered right in two option orders. Against drift, a KL term keeps each stage near its starting model. One pass offers VLAs a more plausible path to System 1. SuperGlance answers visual multiple-choice questions as well as its base VLM does by decoding. It plans in 84 ms of GPU time, six times faster than Qwen-Drive-1.0, and at least as well on NAVSIM, PAI-AV and WOD-E2E. In AlpaSim closed loop, km per at-fault incident holds for both planners up to a 300-ms delay and drops beyond. With GPU time charged as delay, SuperGlance's lead over Qwen-Drive-1.0 grows from 2.8 to 3.9 times. Delayed by 1.2 s, our model falls below Qwen-Drive-1.0 at its own 518 ms.

Models: SuperGlance-9B SuperGlance-9B-Stage1 SuperGlance-9B-Stage2 **Data:** SuperGlance-QA
Code: GitHub **Project Page:** Project Website

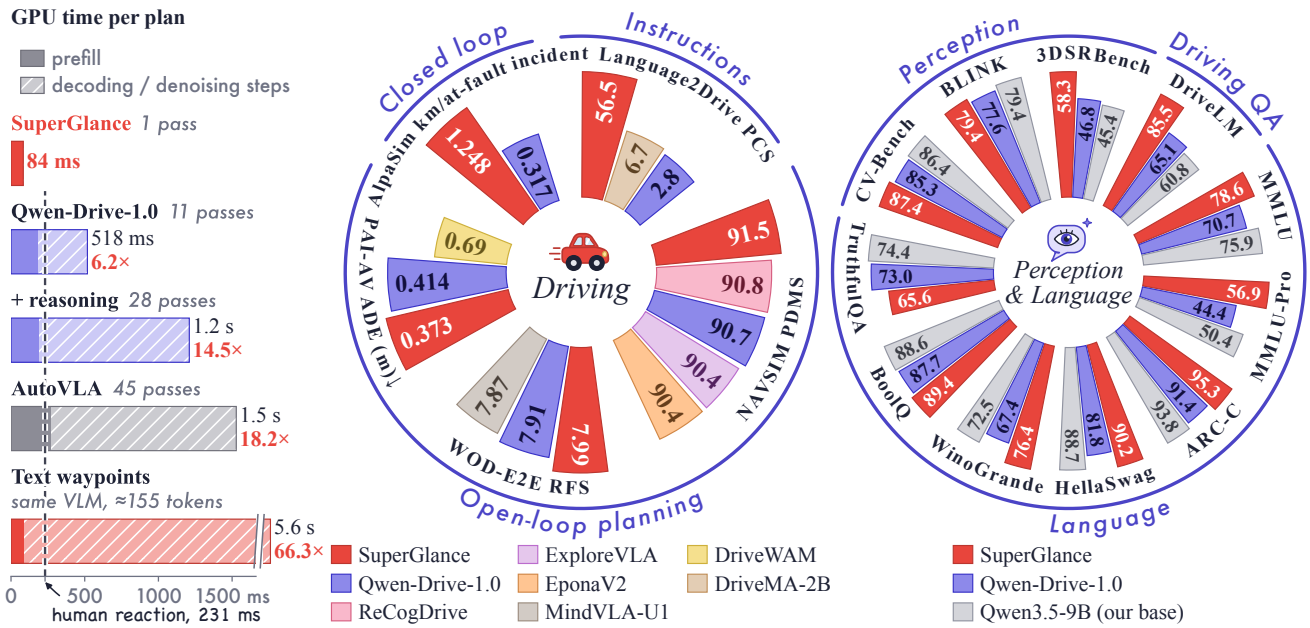


Figure 1 **SuperGlance and benchmark results.** A driving System 1 needs world knowledge and fast reactions. SuperGlance plans in one forward pass, 6.2 times faster than Qwen-Drive-1.0, and leads it on ten of eleven QA and all driving benchmarks. Left: GPU time per plan, charged as delay in AlpaSim. Middle: ours, Qwen-Drive-1.0's best settings [1] and others; Language2Drive as in Tab. 7. Right: ours, Qwen-Drive-1.0 (4B) and our base. Ring bars start above zero; longer ADE bars mean lower error.

1 Introduction

Autonomous driving is turning to vision-language-action models (VLAs), vision-language models (VLMs) trained end to end as driving policies [1–5]. They bring broad visual knowledge. Such a policy reads camera frames and the ego state and outputs a trajectory for the next few seconds. A VLM answers by decoding, and current driving VLAs produce this trajectory step by step as well (Fig. 2a,b): they write waypoints as text or action tokens, one per forward pass [2, 6–8], or denoise it over several steps with a planning expert that reads the VLM's features [1, 5].

Every step is another forward pass, and the passes add up. On one H200 GPU at batch size one, Qwen-Drive-1.0 [1] spends 518 ms of GPU time on a plan over 11 passes, and 1,213 ms when also writing short reasoning (Fig. 1, left). The car keeps moving while the policy thinks, so the plan arrives late (Sec. 4.3). A human driver plans at a glance, with the fast, automatic thinking known as System 1 [9]. Faster designs mostly move planning out of the VLM into a separate planner or smaller student (Sec. 2), leaving the VLM's abilities behind. So far, a fast driving VLA that answers questions either plans outside the VLM or regresses one trajectory from query tokens [10].

Yet before a VLM writes its first token, it has already read the scene into one hidden state. We show that, once trained, this hidden state is enough to drive and to answer questions about what the model sees. VLMs became decoder-only to write free text; a plan is not free text. BERT [11] and GPT [12] classify from one hidden state with a small head, as did early vision-language transformers. We bring this classification approach to driving, with candidate trajectories as the classes, so a plan takes one forward pass.

We present SuperGlance, which turns a decoder-only VLM into an encoder-style driving policy that plans in one forward pass (Fig. 1, left; Fig. 2c). A linear Policy Head on the last hidden state at the answer position, [ANS], scores a fixed set of 5,273 candidate trajectories, most of them cluster centers of human trajectories (Sec. 3.1). As the set is fixed, every candidate's reward in every training scene is computed once, offline. Training has three stages (Fig. 3). Stages 1 and 2 train on multiple-choice questions to keep the VLM's visual abilities in one forward pass and carry them to questions about driving scenes. Stage 3 trains the policy, first toward a supervised target that weights the candidates by reward. This target can still leave much probability on unsafe candidates (Fig. 4), so reinforcement learning (RL) raises the expected reward, computed exactly over all candidates without sampling, and lifts NAVSIM's no-collision and time-to-collision scores. The language-model head is kept and question data stays in training, so the final model still answers questions at almost no cost to planning. With no decoding, denoising or separate planner, the VLM itself plans at a glance, as a human driver does.

Method	Planning paradigm	Sequential steps	Planner in VLM	Keeps language	RL objective
Decoding or denoising VLAs					
AutoVLA [2]	Autoregressive	10 + text	✓	✓	GRPO
Qwen-Drive-1.0 [1]	Flow matching	11 + text	×	✓	GRPO
One-pass or fast VLM planners					
SimLingo [10]	Regression	1 + text	✓	✓	×
AnchorVLA [14]	Anchor + flow	3 + text	×	✓	×
Vocabulary scorers without a VLM					
ZTRS [15]	Scoring	1	×	×	Exact
SuperGlance	Scoring	1	✓	✓	Exact

Table 1 Related driving policies. Text adds a step per token.

We test our final RL checkpoint on NAVSIM, WOD-E2E, PAI-AV and Language2Drive, and in AlpaSim closed loop [13]. With each plan's GPU time charged as delay, it drives 3.9 times as far per at-fault incident as Qwen-Drive-1.0 (Fig. 1). Without delay, it still leads, as does every one-pass variant in our ablations (Sec. 4.3).

- SuperGlance plans from one hidden state of a decoder-only VLM by scoring a fixed candidate set, so RL maximizes the exact expected reward over all candidates.
- Multiple-choice training keeps the VLM's visual abilities in one forward pass and carries them to driving scenes.
- One checkpoint plans in 84 ms, 6.2 times faster than Qwen-Drive-1.0, and at least as well on NAVSIM, WOD-E2E, PAI-AV and AlpaSim; it answers visual multiple-choice questions as well as its base VLM does by decoding.

2 Related Work

2.1 Vision-language models as driving policies

Recent driving VLAs differ mainly in how they emit a trajectory (Tab. 1). AdaThinkDrive [6], Curious-VLA [7] and DriveTeach-VLA [8] write waypoints as text, AutoVLA [2] decodes action tokens, DriveFine [16] decodes bin tokens by masked diffusion, and WAM-Flow [17] refines numeric tokens by discrete flow matching. ReCogDrive [5] and Qwen-Drive-1.0 [1] attach a denoising planner or action expert, and world-model policies [18–21] add future-frame prediction. Most plan over sequential decoding or denoising steps; SuperGlance reads its plan from one hidden state in a single forward pass (Fig. 2).

Fast planning usually moves the plan out of the VLM: dual systems pair it with a conventional or smaller planner [4, 22, 23] or lightweight action expert [24], and distillation trains a student that skips reasoning [25]. CarLLaVA [26], Reasoning-VLA [27] and, for robot actions, OpenVLA-OFT [28] regress from query tokens in one pass; SimLingo [10] does so in a language-capable VLM. AnchorVLA [14] decodes a trajectory-anchor token and refines it with a flow model, ORION [29] a planning token with a generative planner, and FiS-VLA [30] uses a robot VLA's last blocks as a fast policy that still denoises. Unlike dual systems and students, SuperGlance plans with the full VLM.

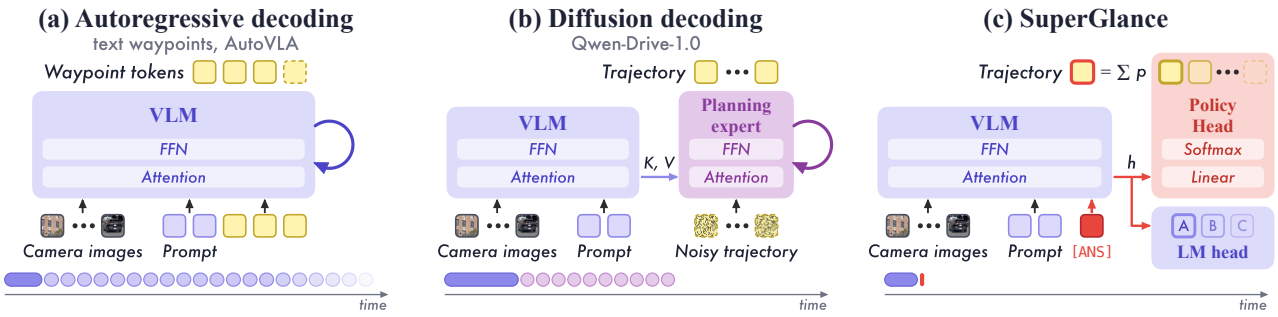


Figure 2 **Three ways a vision-language model produces a plan.** Only (c) has no loop. (a) Autoregressive decoding writes a trajectory one token per forward pass (text waypoints, AutoVLA [2]). (b) Diffusion decoding runs the VLM once, then a planning expert refines a noisy trajectory over several steps (Qwen-Drive-1.0 [1]). (c) SuperGlance reads the last hidden state h once, at [ANS], the prompt's last position. The Policy Head gives a probability (outline width and opacity) to each of 5,273 fixed candidates, and for questions, the LM head gives one to each option letter. Each bottom strip shows one plan's measured GPU time to scale; in (a), our VLM writing text runs off its axis.

Unlike regression and anchor-token policies, it plans by scoring candidates from one hidden state, so its target weights each by reward (Sec. 4.4) and RL uses all of their rewards.

Many driving VLAs add GRPO-style RL [31] over sampled trajectories [1, 2, 5]. Perception-based planners score a fixed trajectory vocabulary [32–35], per-scene candidates [36, 37] or both [38]. ZTRS [15] takes an exact policy gradient over such a vocabulary, and DriveDPO [39] distils imitation and safety scores into a policy aligned by preference optimization. SuperGlance brings the vocabulary and exact, sampling-free RL into a VLM.

2.2 Decision making in vision-language models

Early VLMs such as ViLBERT [40], LXMERT [41] and UNITER [42] followed BERT [11], casting visual question answering (VQA) as classification over frequent answers with a small head per task on a pre-trained encoder. Flamingo [43], BLIP-2 [44] and LLaVA [45] instead write their answers token by token, as do Qwen3-VL [46] and Qwen3.5 [47]. Scoring a fixed answer set takes one pass for any answer length; so does reading option-letter logits for multiple choice [48], though with letter and position biases [49]. SuperGlance brings the classification head back into a generative VLM: its Policy Head scores a fixed set of trajectories as a VQA head scores answers, and the language-model head still answers questions.

3 Method

SuperGlance reads one hidden state through two heads, each in one forward pass: the language-model head answers multiple-choice questions and a linear Policy Head scores a fixed set of candidate trajectories (Fig. 2). Two training stages on multiple-choice questions keep the VLM's visual abilities in one forward pass, and a third trains the policy.

3.1 Architecture

Inputs. Our VLM, Qwen3.5-9B [47], receives the current front-camera frame at 1024×576 and, if available, the three frames taken 1.5, 1.0 and 0.5 s earlier at 544×320: 576 visual tokens for the current frame and 170 per past frame. The prompt gives the ego speed, acceleration, past 1.5 s of positions, navigation command and any instruction, and ends at the answer position, [ANS]. No detector or map output enters the prompt; the scene comes from the frames alone.

Candidates. Each candidate is a 4-s trajectory of 8 poses (position and heading) at 2 Hz in the ego frame. Of the $K = 5{,}273$ candidates, 4,096 are k-means centers of human trajectories in the 92,339 NAVSIM training scenes, and one is stationary. NAVSIM rarely reaches 13 m/s, so 1,024 more are k-means centers of PAI-AV driving at that speed or faster. Clustering misses stops and curves that are rare in human driving, so 152 more are built by rule (appendix). The best clustered candidate per scene reaches a mean PDM score (PDMS) of 99.84 on 10,549 held-out NAVSIM training scenes and a mean rater feedback score (RFS) of 9.55 on the rated WOD-E2E validation frames.

Policy Head. The Policy Head is a linear layer with weights $W \in \mathbb{R}^{K\times d}$ and bias $b \in \mathbb{R}^K$, one row per candidate, on the last hidden state $h \in \mathbb{R}^d$ at [ANS], $d = 4{,}096$. For scene inputs x, it gives the policy and the plan

$$\begin{aligned} p_\theta(k \mid x) &= \mathrm{softmax}(Wh + b)_k, \\ \bar{a}(x) &= \textstyle\sum_{k\in T(x)} w_k\, a_k, \quad w_k \propto p_\theta(k \mid x)^{1/\tau}, \end{aligned} \tag{1}$$

where a_k is candidate k's trajectory, $T(x)$ holds the 64 most probable candidates and the temperature is $\tau = 0.2$. This top-64 mean adds 1.21 PDMS on navtest over the most probable candidate, and 0.23 after RL. In AlpaSim, a low-level controller executes every method's plan as in NAVSIM scoring: NAVSIM's PDM simulator [50, 51], an LQR tracker on a kinematic bicycle model. It runs the plan from the ego's



Figure 3 **Training data of each stage.** Every stage trains an answer read in one forward pass: a distribution over option letters, and in stage 3 also over candidates. Left: held-out samples where Qwen3.5-9B is wrong and the model after that stage is right. A tick marks our pick, a cross Qwen3.5-9B's, and questions are lightly abridged. In the stage-3 row, the four paths are our supervised model's pick among its 5,121 candidates and three low-reward candidates, each with its score. Right: each stage's sources with shares (rings) and counts, and the 5,273 candidates seen from above, coloured by origin (Sec. 3.1).

current speed and acceleration and the steering angle reached while running the previous plan, and passes 40 poses, 0.1 s apart, to AlpaSim's controller. The language-model head is unchanged; for multiple choice, its answer distribution p is the softmax over the option-letter logits at [ANS].

3.2 Data

QA data. Stage 1 uses 1.16M questions from visual and text datasets [52–57] and stage 2 uses 124,378 driving questions from DriveLM-nuScenes [3], all recast as multiple choice with one gold option letter (Fig. 3; appendix).

Rewards. Because the candidates are fixed, every candidate's reward $r(x, k)$ in every training scene x is computed once before training. On NAVSIM [50], it is the PDMS times the driving-direction compliance (DDC), both in $[0, 1]$. A candidate that collides, leaves the drivable area or violates the time-to-collision margin counts as unsafe, so it loses a further 10 and ranks below every safe candidate. On PAI-AV [58], it is -0.2 times the average displacement error (ADE, in meters) to the logged future, and on the rated WOD-E2E frames [59] it is the RFS divided by 10.

Rollout rows. In closed loop, a policy drifts into states that logged driving never shows. To learn to recover, stage 3 also trains on 46,583 frames, 2.1% of its rows (examples), rendered along AlpaSim rollouts of earlier models on 492 scenes outside the evaluation scenarios. Their target trajectory rejoins the human path in about 3 s, never ahead of the human, and slows behind a lead vehicle.

3.3 Training

Each stage trains rank-32 LoRA adapters [60] on the language model, merged before the next, and a KL term keeps each stage near its starting model, so that later stages keep what earlier ones learned.

Stage 1: general QA. Read in one pass, a VLM's option letter is biased toward some letters and positions [49], so stage 1 trains it to follow the content. Each question with gold letter y has a twin with shifted options and gold y'. One letter is drawn from each order's distribution, p and p', and the pair hits only if both are gold, so an order-dependent answer earns no credit (Fig. 5). The reward is one minus the Brier score [61] of the pair distribution $p \otimes p'$, and 32 pairs estimate its gradient without bias (appendix).

Stage 2: driving QA. Stage 2 repeats stage 1 on DriveLM questions about objects, their relations and the ego vehicle, carrying the VLM's abilities to driving scenes. Stage-1 questions stay in its batches to keep the general abilities.

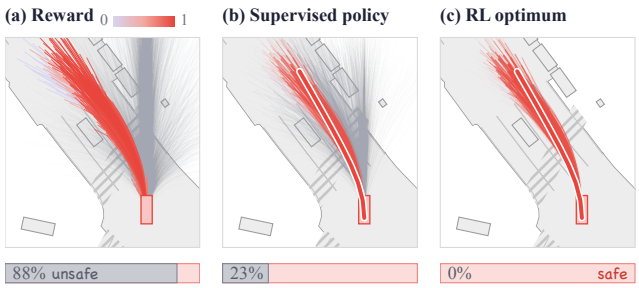


Figure 4 From rewards to a policy on one navtest scene. (a) Each candidate is colored by its reward; unsafe ones are gray (Sec. 3.2). (b) In the supervised policy, width and opacity grow with probability, and unsafe candidates hold nearly a quarter of it. (c) The RL optimum, computed from (a) and (b) by Eq. (4) without the RL model, gives them none. Bars give the unsafe share of candidates (a) and of probability (b, c).

Stage 3: policy training. Stage 3 trains the policy, first toward a supervised target $q(k \mid x) \propto \exp\big(s(x,k)/t\big)$ that spreads probability over the candidates by a score $s(x,k)$:

$$\mathcal{L}_{\mathrm{SFT}} = -\sum_k q(k \mid x) \log p_\theta(k \mid x). \qquad (2)$$

On NAVSIM, s is PDMS$\times$DDC minus 0.1 times the ADE in meters, with $t = 0.1$, so the target keeps its mass on safe candidates and, among them, favors those near the human trajectory. On PAI-AV and WOD-E2E, s is the negative ADE, with $t = 0.5$ m and 1 m; a last phase uses WOD-E2E frames around the rated ones, with an RFS-based score (appendix). Stage-1 and stage-2 questions stay in the batches so that the policy keeps answering questions (Sec. 4.2).

The 152 rule-built candidates are added after supervised training, so before RL the Policy Head is refit to all 5,273 on the supervised model's features (appendix). The resulting policy, p_{SFT}, still leaves probability on unsafe candidates (Fig. 4b), so RL maximizes the expected reward with a KL term that keeps the policy close to p_{SFT}:

$$\mathbb{E}_x\big[\textstyle\sum_k p_\theta(k \mid x)\, r(x,k) - \beta\, \mathrm{KL}(p_\theta \,\|\, p_{\mathrm{SFT}})\big]. \qquad (3)$$

The sum runs over all K candidates, so the expectation and its gradient are exact; no trajectory is sampled or simulated (appendix). As in KL-regularized RL [62, 63], the policy that maximizes Eq. (3) is, in each scene,

$$p^\star(k \mid x) \propto p_{\mathrm{SFT}}(k \mid x)\, \exp\big(r(x,k)/\beta\big). \qquad (4)$$

It reweights p_{SFT} by reward, and β sets how far it moves from p_{SFT} (Fig. 4c); we use $\beta = 0.3$.

4 Experiments

4.1 Setup

Benchmarks. We evaluate driving broadly, on five benchmarks: open-loop planning on NAVSIM, WOD-E2E and PAI-AV, closed-loop driving in AlpaSim and instruction following on Language2Drive (Tab. 7). **NAVSIM** [50] v1.1 navtest has 12,146 scenes scored by PDMS, which multiplies no-collision (NC) and drivable-area compliance (DAC) by a weighted sum of ego progress (EP), time-to-collision (TTC) and comfort (Comf.). **WOD-E2E** [59] covers long-tail scenes; we report RFS and 3- and 5-s ADE on its test server. **PAI-AV** [58] open loop is scored by 3- and 5-s ADE on 700 frames from 140 held-out clips. **AlpaSim** [13] runs closed loop on the 916 scenarios of PAI-AV-NuRec [74, 75] v26.01, and the policy plans at every call. We report collision and off-road rates, progress, share of scenarios passed and distance driven per incident (collision or off-road event). At-fault rates leave out hits on the ego's rear. A scenario passes if it has no at-fault collision, off-road event or lateral-corridor exit. Its score is progress along the human path divided by 0.8, capped at 1, or zero if it fails.

Baselines. We run Qwen-Drive-1.0's two released planners on PAI-AV and in AlpaSim, executed like ours.

	Qwen3.5-9B			SuperGlance after stage		
Benchmark	letter	decoded	Qwen-Drive-1.0	1	2	3
Vision						
CV-Bench [64]	86.36	86.47	85.29	87.60	87.66	87.40
BLINK [65]	79.37	76.62	77.60	84.68	81.34	79.37
3DSRBench [66]	45.39	57.09	46.81	60.27	60.50	58.29
Language						
MMLU [67]	75.91	85.21	70.69	79.79	79.63	78.61
MMLU-Pro [68]	50.41	73.93	44.35	59.03	57.98	56.85
ARC-C [69]	93.77	96.16	91.38	95.39	95.14	95.31
HellaSwag [70]	88.67	89.49	81.83	91.00	91.04	90.15
WinoGrande [71]	72.53	85.64	67.40	76.40	77.43	76.40
BoolQ [72]	88.62	88.29	87.65	89.72	89.48	89.42
TruthfulQA [73]	74.42	76.25	72.95	67.07	66.22	65.61
Driving						
DriveLM [3]	60.81	60.95	65.06	73.55	84.33	85.51

Table 2 Question answering. Accuracy (%) after each training stage of SuperGlance; all models are scored by us with the same prompts and option-letter readout. Shading deepens from the worst to the best score, by row here and by column below. The decoded column reads Qwen3.5-9B's written replies (thinking off). Stage 3 is our final RL model. For BLINK we use counting, object localization, relative depth and spatial relation. DriveLM (our scene-split test questions) and 3DSRBench count an answer only if it holds in every cyclic option order. Only DriveLM is in our training data.

4.2 Efficiency and question answering

Generation dominates each plan's GPU time on one H200 (Fig. 1, left, and Fig. 6, top; protocol in the appendix): a flow step of Qwen-Drive-1.0 or a decoded token of our VLM takes 33–35 ms, so Qwen-Drive-1.0's ten flow steps alone take four times our whole 84 ms pass, and writing the plan as text takes our VLM 66 times as long. Reading alone takes Qwen-Drive-1.0 187 ms, over twice our whole pass, as it reads nearly four times as many image tokens.

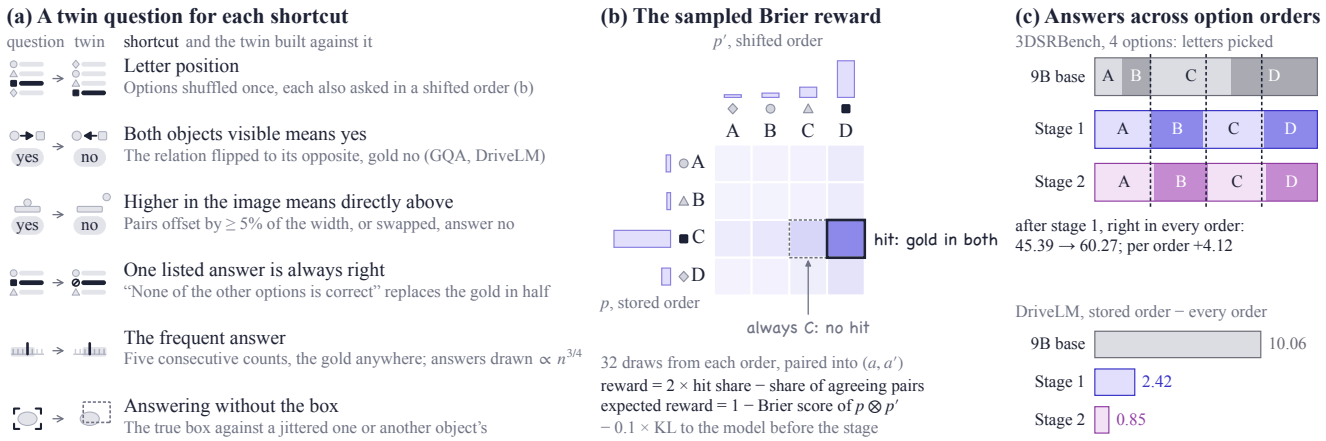


Figure 5 **Questions built against shortcuts in stages 1 and 2.** A one-pass letter can be right by position alone. (a) A shortcut answers a question right (left) and its twin wrong (right); gold is dark. (b) Rows use the stored order (gold at C) and columns the shifted one (gold at D); bars show illustrative letter probabilities, shades pair probabilities. (c) Top: letter shares on 3DSRBench's 4-option questions, with gold spread evenly (dashed: 25% each); Qwen3.5-9B picks C or D for 75%, by position, and after stage 2 each letter takes 23–27%.

Multiple-choice training keeps the VLM's visual abilities in one pass (Tab. 2). On 3DSRBench, the base model's letter trails its own decoded replies by 11.70 points, mostly through position bias rather than missing knowledge. Stage 1 closes the gap by answering consistently across option orders (Fig. 5c). After all three stages, our letter is level with or above Qwen3.5-9B's decoded replies (thinking off) on every vision benchmark and 24.56 points above them on DriveLM, which we train on. It is at or above Qwen3.5-9B's letter on every benchmark but TruthfulQA, and policy training in stage 3 moves no score by more than 2.21 points. On language, Qwen3.5-9B's written replies lead our letter by up to 17.08 points by reasoning before answering; answering first, they stay within 1.4 points of its letter (appendix).

Planning and answering. Question data keeps the VLM's world knowledge and costs planning almost nothing (appendix Tab. 19): without it, the model is level on navtest and AlpaSim and 0.01 m lower in PAI-AV's 3-s ADE, but falls below Qwen3.5-9B's letter on all eleven benchmarks. Skipping only stages 1 and 2 leaves planning level and costs DriveLM and 3DSRBench 3.20 and 4.36 points, a tenth and a third of the loss without question data.

4.3 Driving results

AlpaSim. Time is distance in closed loop (Tab. 3 and Fig. 6): with each plan's GPU time as its delay, our RL model drives 3.9 times as far per at-fault incident as Qwen-Drive-1.0 (95% interval 3.3 to 4.7) and passes 72.2% of scenarios, against at most 36.6%. Delay costs distance once it passes 300 ms: up to that lag, both planners keep their distance within 8%; at Qwen-Drive-1.0's own 518 ms, it loses a factor of 1.5 and ours 1.2, and at 1.2 s ours loses 5.1. Without delay, our supervised model drives 2.6 and 3.1 times as far as

Method	RL	Coll.↓	Off-road↓	Prog.↑	km / incident↑	Pass↑
Each plan's GPU time as delay						
Qwen-Drive-1.0, 518 ms		32.6/18.1	19.3	67.9	0.229/0.317	36.6
+ reasoning, 1.2 s		26.7/21.0	36.3	51.8	0.145/0.158	9.3
SuperGlance, 84 ms	✓	10.9/6.8	7.5	88.5	0.970/1.248	72.2
No delay (simulator default)						
Qwen-Drive-1.0 [1]		32.8/14.4	15.4	71.1	0.284/0.461	58.7
Qwen-Drive-1.0	✓	43.6/15.6	13.1	57.3	0.194/0.383	53.6
SuperGlance		11.2/6.0	8.8	87.9	0.886/1.199	72.1
SuperGlance	✓	10.6/5.9	7.6	88.6	0.975/1.318	73.5

Table 3 **AlpaSim closed loop.** Top: each plan's GPU time acts as its delay, rounded up to a lag of 600 ms, 1.2 s or 100 ms (appendix Sec. D); bottom: no delay. Qwen-Drive-1.0 plans with reasoning except in its 518 ms row. Coll. and km / incident: all / at-fault; Coll., off-road, progress and pass in %.

Qwen-Drive-1.0's two planners, and every one-pass variant in appendix Tab. 19 drives at least 1.7 times as far. RL keeps the mean scenario score level (+0.017, 95% interval −0.004 to +0.038); appendix Fig. 9 compares our driving with that of Qwen-Drive-1.0 without delay.

NAVSIM. Supervised training alone puts SuperGlance above every other method in Tab. 4, owing to its reward target and the top-64 mean (Secs. 3.1 and 4.4). RL then trades progress for safety: the exact expectation (Eq. (3)) raises NC and TTC (appendix Tab. 20), and the RL model reaches 91.5 PDMS, 0.8 above Qwen-Drive-1.0 after RL.

WOD-E2E and PAI-AV. No WOD-E2E trajectory is among the candidates, yet our plan scores 7.99 RFS on its long-tail test scenes, 0.08 above Qwen-Drive-1.0 after RL (Tab. 5). On PAI-AV, our single plan lands closer to the logged future than either Qwen-Drive-1.0 planner at both horizons, even when that planner keeps the best of six samples, and the best of our six most probable candidates is

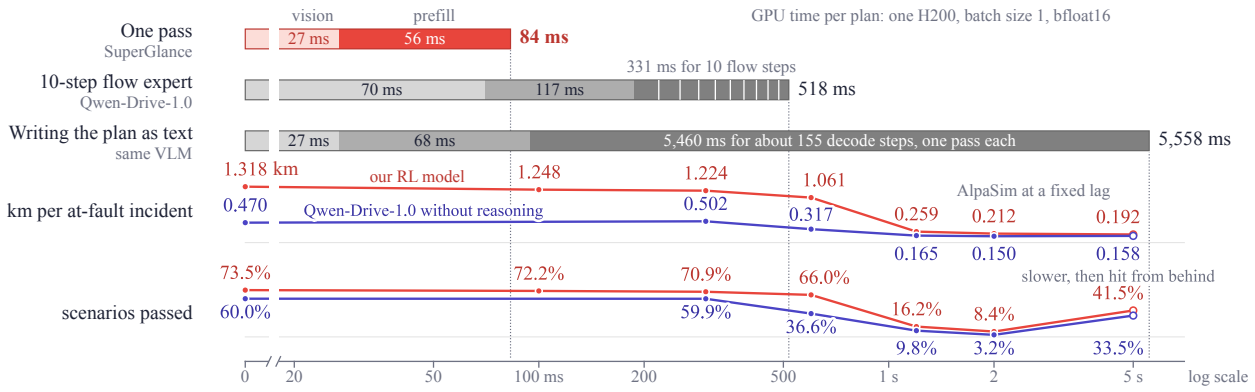


Figure 6 **Time per plan and distance in closed loop.** Top: a plan takes 84 ms of GPU time in our single pass, 518 ms in Qwen-Drive-1.0 with 10 flow steps and 5,558 ms when our VLM writes it as text. Bottom: in AlpaSim, each time (dotted) becomes the next 100-ms lag, and both planners keep their distance per at-fault incident within 8% up to 300 ms. At 600 ms, Qwen-Drive-1.0 falls to 0.317 km and ours to 1.061 km; at 1.2 s ours falls to 0.259 km, below Qwen-Drive-1.0 at its own 518 ms. At 5 s every plan arrives after its last pose (hollow): both planners drive more slowly and vehicles behind hit them, which is not at fault and ends the scenario as a pass.

Method	NC↑	DAC↑	EP↑	TTC↑	Comf.↑	PDMS↑
Supervised						
TransFuser [76]	97.7	92.8	79.2	92.8	100.0	84.0
DRAMA [77]	98.0	93.1	80.1	94.8	100.0	85.5
Hydra-MDP [33]	98.3	96.0	78.7	94.6	100.0	86.5
DiffusionDrive [78]	98.2	96.2	82.2	94.7	100.0	88.1
Epona [18]	97.9	95.1	80.4	93.8	99.9	86.2
ReCogDrive [5]	98.1	94.7	80.9	94.2	100.0	86.5
AutoVLA [2]	96.9	92.4	75.8	88.1	99.9	80.5
Qwen-Drive-1.0 [1]	98.2	96.4	82.0	94.4	100.0	87.8
+ reasoning	98.4	96.6	82.4	94.7	100.0	88.2
SuperGlance	98.7	98.3	86.2	95.6	100.0	91.2
RL						
ReCogDrive	97.9	97.3	87.3	94.9	100.0	90.8
AutoVLA	98.4	95.6	81.9	98.0	99.9	89.1
ExploreVLA [79]	98.8	98.4	83.5	96.5	99.9	90.4
EponaV2 [19]	98.6	97.9	84.8	95.7	100.0	90.4
Qwen-Drive-1.0	98.6	98.2	84.8	95.9	100.0	90.7
SuperGlance	99.3	98.7	84.0	97.6	100.0	91.5

Table 4 **NAVSIM v1.1 navtest.** One plan per scene; shading ranks the methods within each group, and Comf. is unshaded.

Method	RL	3-s ADE↓	5-s ADE↓	RFS↑
MindVLA-U1 [80]		1.16	2.67	7.77
AutoVLA [2]	✓	1.35	2.96	7.56
MindVLA-U1	✓	1.09	2.66	7.87
Qwen-Drive-1.0 + reasoning [1]		1.19	2.65	7.78
Qwen-Drive-1.0 + reasoning	✓	1.19	2.67	7.91
SuperGlance	✓	1.19	2.55	7.99

Table 5 **WOD-E2E test split.** ADE in metres; our 4-s plans are extended to 5 s at constant velocity.

closer still (Tab. 6). Most of its gain over our supervised model comes from the 152 candidates built by rule for braking and medium-speed curves (appendix).

Manoeuvre and choice. One pass settles the manoeuvre and leaves the choice within it. On navtest, our RL model's most probable candidate takes the driver's manoeuvre in

		One plan		Best of six	
Method	RL	3 s	5 s	3 s	5 s
DriveWAM [20]†		0.69	–	0.38	–
Qwen-Drive-1.0 + reasoning [1]		0.414	1.205	0.385	1.095
Qwen-Drive-1.0 + reasoning	✓	0.454	1.231	0.415	1.102
SuperGlance		0.394	1.109	0.284	0.821
SuperGlance	✓	0.373	1.076	0.271	0.777

Table 6 **PAI-AV open loop.** ADE (m)↓ on 700 held-out frames. One plan: our top-64 mean and Qwen-Drive-1.0's first sample; best of six: our six most probable candidates and its six samples. We extend our 4-s plans to 5 s at constant velocity; †: reproduced on these frames in the Qwen-Drive-1.0 paper; –: not evaluated.

Method	SemSISR↑	SemCFPS↑	SCS↑	PCS↑
Qwen-Drive-1.0 (4B) [1]	15.8	0.0	25.3	2.8
AutoVLA [2]	18.8	0.0	30.2	5.4
DriveMA-2B [81]	20.4	0.0	30.6	6.7
ReCogDrive [5]	13.2	0.0	22.5	2.8
OneVL [82]	20.7	0.0	30.5	4.6
SuperGlance, no instruction	11.8	0.0	26.2	6.2
SuperGlance, no instruction rows	15.8	0.0	27.2	4.7
SuperGlance	40.5	21.1	65.0	56.5

Table 7 **Language2Drive, NAVSIM track.** 38 instruction pairs in four wordings each, scored with the benchmark's current scorer, others from their predictions. SemSISR, SemCFPS: single instructions and pairs followed (%); SCS, PCS: graded scores.

89% of scenes (the driver goes straight in 56%; appendix Fig. 10a), and its six most probable candidates end 1.3 m apart at 4 s (median of mean pairwise distances), against 33 m for six random ones. Within the manoeuvre, the best of the six still scores 94.80 PDMS against 91.28 for the most probable one (appendix Fig. 10b).

Instructions. Our policy follows instruction pairs that the five VLAs in Language2Drive [83] miss. The benchmark

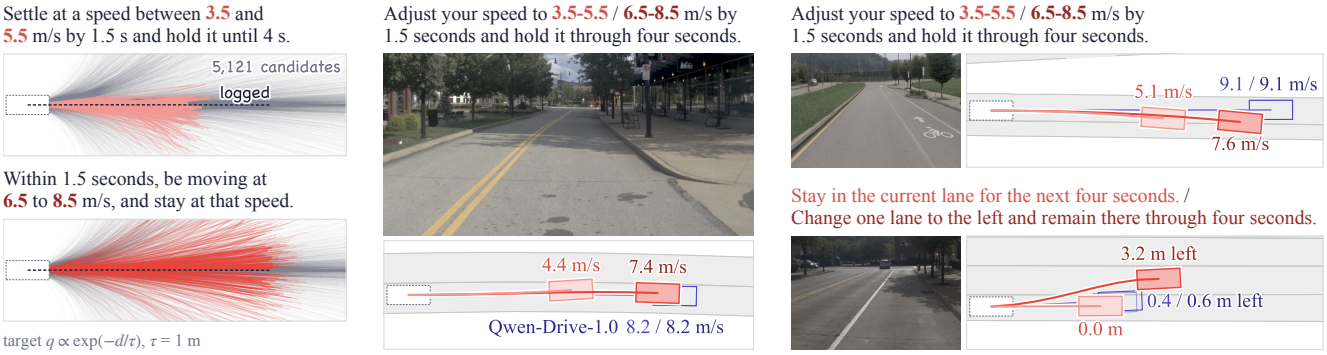


Figure 7 **Language steers the plan.** Each instruction selects its own plan from the same candidates. Left: two instructions on a training scene, each with the candidates that follow it (red); line weight is the target $q \propto \exp(-d/\tau)$, $\tau = 1$ m, with d the distance to the logged future in what the instruction leaves free (across the path for a speed instruction). Right: plans for both instructions (light, dark) in three Language2Drive scenes, boxes at 4 s; ours (red) change with the instruction, Qwen-Drive-1.0's (purple) barely.

		Open loop	Closed loop (AlpaSim)	
Output	Target	PDMS↑	km / incident↑	Pass (%)↑
Text, decoded	Imitation	87.2	–	–
Linear regression	Imitation	87.8	1.360	70.3
Policy Head	Imitation	82.0	1.199	71.0
Policy Head	Reward	91.3	1.486	73.4
Qwen-Drive-1.0 + reasoning [1]		88.2	0.461	58.7

Table 8 **Output and target.** Variants of our supervised model before its last phase, with Qwen-Drive-1.0 for reference. Imitation: the target follows the human trajectory; –: not evaluated (writing the plan as text takes 5,558 ms, longer than the 4 s it covers). PDMS is on navtest; AlpaSim without delay, km per at-fault incident.

gives a policy two instructions per scene, such as two speed bands, and checks whether each plan follows its own. With its current scorer, our RL model follows both in 21.1% of pairs, against 0.0% for each of the five and for our model without the instruction (Tab. 7 and Fig. 7). As in language models, each ability comes with rows of its kind: DriveLM questions lift DriveLM (Tab. 2), and instruction rows in this style, on training scenes only, bring instruction following; without them, our supervised model follows no pair.

History and rollout rows. Closed loop shows what navtest does not. For our supervised model before its last phase, dropping the history frames doubles the AlpaSim scenarios with an at-fault collision, from 37 to 73, and dropping the rollout rows raises them to 88, while navtest moves by under half a point (appendix Tab. 19). Without history, front hits on stopped vehicles rise from 10 to 44. Behavior cloning on histories is reported to do the reverse, better offline and worse in closed loop, by copying past actions [84, 85].

4.4 Ablations on output and target

The hidden state before the first token is enough to plan: given the same data, inputs and imitation target, linear regression on it plans at least as well as the same VLM writing the plan as text, which takes 66 times the GPU time (Tab. 8; level on PAI-AV and WOD-E2E, appendix). The regression also has the lowest single-plan PAI-AV ADE of any model we ran, 0.320 m at 3 s (appendix Tab. 19). A fixed set pays off through its target. Under imitation it trails the regression on navtest, partly because even the candidate nearest the human trajectory reaches only 85.7 PDMS on held-out training scenes. The reward target lifts the Policy Head to 91.3 and raises AlpaSim's mean scenario score.

5 Conclusion

We find that before a VLM writes its first token, it has already read the scene into one hidden state, which, once trained, is enough to drive and to answer questions about what the model sees. SuperGlance reads it to turn a decoder-only VLM into an encoder-style driving policy whose fixed candidates let targets weight plans by reward, RL maximize the exact expected reward and instructions pick among them. Questions built against shortcuts, each answered in both option orders, keep the VLM's visual abilities in one pass. One pass also reacts faster, and time is distance: with each plan's GPU time as delay, our model drives 3.9 times as far per at-fault incident in closed loop as Qwen-Drive-1.0. One model answers questions on eleven benchmarks and plans in 84 ms of GPU time: a path to System 1 in the VLM.

LLM Usage Statement

We used Claude Code (Anthropic) extensively in this work: to search related work, propose and discuss ideas, write training, evaluation and figure code, launch and monitor experiments, analyse results, and draft and edit the text of this paper; Gemini (Google) also suggested wording edits. The authors directed the research, checked the results and the text, and take full responsibility for them.

References

[1] Xin Zhou, Zongchuang Zhao, Zhibo Yang, et al. Qwen-Drive-1.0: An Initial Step towards a Vision-Language Foundation Model for Autonomous Driving. *arXiv preprint arXiv:2609.00111*, 2026.

[2] Zewei Zhou, Tianhui Cai, Seth Z. Zhao, et al. AutoVLA: A Vision-Language-Action Model for End-to-End Autonomous Driving with Adaptive Reasoning and Reinforcement Fine-Tuning. In *NeurIPS*, 2025.

[3] Chonghao Sima, Katrin Renz, Kashyap Chitta, et al. DriveLM: Driving with Graph Visual Question Answering. In *ECCV*, 2024.

[4] Xiaoyu Tian, Junru Gu, Bailin Li, et al. DriveVLM: The Convergence of Autonomous Driving and Large Vision-Language Models. In *CoRL*, 2024.

[5] Yongkang Li, Kaixin Xiong, Xiangyu Guo, et al. ReCogDrive: A Reinforced Cognitive Framework for End-to-End Autonomous Driving. In *ICLR*, 2026.

[6] Yuechen Luo, Fang Li, Shaoqing Xu, et al. AdaThinkDrive: Adaptive Thinking via Reinforcement Learning for Autonomous Driving. In *ICRA*, 2026.

[7] Canyu Chen, Yuguang Yang, Zhewen Tan, et al. Devil is in Narrow Policy: Unleashing Exploration in Driving VLA Models. In *Findings of CVPR*, 2026.

[8] Yuguang Yang, Canyu Chen, Zhewen Tan, et al. Teaching Vision-Language-Action Models What to See and Where to Look. In *ECCV*, 2026.

[9] Daniel Kahneman. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, 2011.

[10] Katrin Renz, Long Chen, Elahe Arani, and Oleg Sinavski. SimLingo: Vision-only closed-loop autonomous driving with language-action alignment. In *CVPR*, 2025.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[12] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018.

[13] NVIDIA, Yulong Cao, Riccardo de Lutio, et al. AlpaSim: A modular, lightweight, and data-driven research simulator for autonomous driving. https://github.com/NVlabs/alpasim, 2025.

[14] Qi Liu, Yabei Li, Hongsong Wang, Heng Zhang, and Lei He. AnchorVLA: Bridging Discrete Decisions and Continuous Trajectories for Vision-Language-Action Planning. *arXiv preprint arXiv:2607.03182*, 2026.

[15] Zhenxin Li, Nadine Chang, Wenhao Yao, et al. Zero-Human Demonstration End-to-end Autonomous Driving with Trajectory Scorer. In *ECCV*, 2026.

[16] Chenxu Dang, Sining Ang, Yongkang Li, et al. DriveFine: Refining-Augmented Masked Diffusion VLA for Efficient and Robust Driving. In *ECCV*, 2026.

[17] Yifang Xu, Jiahao Cui, Zhihao Zhu, et al. WAM-Flow: Parallel Coarse-to-Fine Motion Planning via Discrete Flow Matching for Autonomous Driving. In *CVPR*, 2026.

[18] Kaiwen Zhang, Zhenyu Tang, Xiaotao Hu, et al. Epona: Autoregressive diffusion world model for autonomous driving. In *ICCV*, 2025.

[19] Jiawei Xu, Zhizhou Zhong, Zhijian Shu, et al. EponaV2: Driving world model with comprehensive future reasoning. *arXiv preprint arXiv:2605.14696*, 2026.

[20] Chen Shi, Jinrui Xu, Shaoshuai Shi, Kehua Sheng, Bo Zhang, and Li Jiang. DriveWAM: Video generative priors enable scalable world-action modeling for autonomous driving. *arXiv preprint arXiv:2605.28544*, 2026.

[21] Zongchuang Zhao, Xin Zhou, Tianyang Xu, et al. SimWAM: A simple world action model for end-to-end autonomous driving. *arXiv preprint arXiv:2608.07468*, 2026.

[22] Jianbiao Mei, Yukai Ma, Xuemeng Yang, et al. Continuously learning, adapting, and improving: A dual-process approach to autonomous driving. In *NeurIPS*, 2024.

[23] Ruifei Zhang, Junlin Xie, Wei Zhang, et al. AdaDrive: Self-adaptive slow-fast system for language-grounded autonomous driving. In *ICCV*, 2025.

[24] Yingyan Li, Shuyao Shang, Weisong Liu, et al. DriveVLA-W0: World Models Amplify Data Scaling Law in Autonomous Driving. In *ICLR*, 2026.

[25] Zhaohong Liu, Hao Ye, Xianlin Zhang, and Mengshi Qi. CritiqueDriveVLM: From verifier-guided reinforcement learning to latent thought distillation for autonomous driving. In *ECCV*, 2026.

[26] Katrin Renz, Long Chen, Ana-Maria Marcu, et al. CarLLaVA: Vision language models for camera-only closed-loop driving. *arXiv preprint arXiv:2406.10165*, 2024.

[27] Dapeng Zhang, Zhenlong Yuan, Zhangquan Chen, et al. Reasoning-VLA: An efficient and spatial-guided general vision-language-action reasoning model for autonomous driving. In *ICML*, 2026.

[28] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *RSS*, 2025.

[29] Haoyu Fu, Diankun Zhang, Zongchuang Zhao, et al. ORION: A holistic end-to-end autonomous driving framework by vision-language instructed action generation. In *ICCV*, 2025.

[30] Hao Chen, Jiaming Liu, Chenyang Gu, et al. Fast-in-slow: A dual-system VLA model unifying fast manipulation within slow reasoning. In *NeurIPS*, 2025.

[31] Zhihong Shao, Peiyi Wang, Qihao Zhu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[32] Bo Jiang, Shaoyu Chen, Hao Gao, et al. VADv2: End-to-End Vectorized Autonomous Driving via Probabilistic Planning. In *ICLR*, 2026.

[33] Zhenxin Li, Kailin Li, Shihao Wang, et al. Hydra-MDP: End-to-end Multimodal Planning with Multi-target Hydra-Distillation. *arXiv preprint arXiv:2406.06978*, 2024.

[34] Wenhao Yao, Zhenxin Li, Shiyi Lan, et al. DriveSuprim: Towards Precise Trajectory Selection for End-to-End Planning. In *AAAI*, 2026.

[35] Wenchao Sun, Xuewu Lin, Keyu Chen, et al. SparseDriveV2: Scoring is All You Need for End-to-End Autonomous Driving. In *ECCV*, 2026.

[36] Ellington Kirby, Alexandre Boulch, Yihong Xu, et al. Driving on Registers. In *CVPR*, 2026.

[37] Jieyuan Pei, Meiyi Lu, Sining Ang, et al. World4Scorer: Outcome-Grounded World Modeling for Autonomous Driving. *arXiv preprint arXiv:2609.36438*, 2026.

[38] Zhenxin Li, Wenhao Yao, Zi Wang, et al. Generalized Trajectory Scoring for End-to-end Multimodal Planning. *arXiv preprint arXiv:2506.06664*, 2025.

[39] Shuyao Shang, Yuntao Chen, Yuqi Wang, Yingyan Li, and Zhaoxiang Zhang. DriveDPO: Policy learning via safety DPO for end-to-end autonomous driving. In *NeurIPS*, 2025.

[40] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. ViLBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *NeurIPS*, 2019.

[41] Hao Tan and Mohit Bansal. LXMERT: Learning cross-modality encoder representations from transformers. In *EMNLP-IJCNLP*, 2019.

[42] Yen-Chun Chen, Linjie Li, Licheng Yu, et al. UNITER: UNiversal image-TExt representation learning. In *ECCV*, 2020.

[43] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, et al. Flamingo: a visual language model for few-shot learning. In *NeurIPS*, 2022.

[44] 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.

[45] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023.

[46] Shuai Bai, Yuxuan Cai, Ruizhe Chen, et al. Qwen3-VL Technical Report. *arXiv preprint arXiv:2511.21631*, 2025.

[47] Qwen Team. Qwen3.5: Towards native multimodal agents. https://qwen.ai/blog?id=qwen3.5, 2026.

[48] Revanth Gangi Reddy, JaeHyeok Doo, Yifei Xu, et al. FIRST: Faster Improved Listwise Reranking with Single Token Decoding. In *EMNLP*, 2024.

[49] Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. Large Language Models Are Not Robust Multiple Choice Selectors. In *ICLR*, 2024.

[50] Daniel Dauner, Marcel Hallgarten, Tianyu Li, et al. NAVSIM: Data-Driven Non-Reactive Autonomous Vehicle Simulation and Benchmarking. In *NeurIPS*, 2024.

[51] Daniel Dauner, Marcel Hallgarten, Andreas Geiger, and Kashyap Chitta. Parting with misconceptions about learning-based vehicle motion planning. In *CoRL*, 2023.

[52] Drew A. Hudson and Christopher D. Manning. GQA: A New Dataset for Real-World Visual Reasoning and Compositional Question Answering. In *CVPR*, 2019.

[53] Arijit Ray, Jiafei Duan, Ellis Brown, et al. SAT: Dynamic Spatial Aptitude Training for Multimodal Language Models. In *COLM*, 2025.

[54] Manoj Acharya, Kushal Kafle, and Christopher Kanan. TallyQA: Answering Complex Counting Questions. In *AAAI*, 2019.

[55] Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A Diagram Is Worth A Dozen Images. In *ECCV*, 2016.

[56] Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A Benchmark for Question Answering about Charts with Visual and Logical Reasoning. In *Findings of ACL*, 2022.

[57] Amanpreet Singh, Vivek Natarajan, Meet Shah, et al. Towards VQA Models That Can Read. In *CVPR*, 2019.

[58] NVIDIA. Physical AI autonomous vehicles dataset. https://huggingface.co/datasets/nvidia/PhysicalAI-Autonomous-Vehicles, 2025.

[59] Runsheng Xu, Hubert Lin, Wonseok Jeon, et al. WOD-E2E: Waymo open dataset for end-to-end driving in challenging long-tail scenarios. In *CVPR*, 2026.

[60] Edward J. Hu, Yelong Shen, Phillip Wallis, et al. LoRA: Low-Rank Adaptation of Large Language Models. In *ICLR*, 2022.

[61] Glenn W. Brier. Verification of Forecasts Expressed in Terms of Probability. *Monthly Weather Review*, 78(1):1–3, 1950.

[62] Tomasz Korbak, Ethan Perez, and Christopher L. Buckley. RL with KL penalties is better viewed as Bayesian inference. In *Findings of EMNLP*, 2022.

[63] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*, 2023.

[64] Shengbang Tong, Ellis Brown, Penghao Wu, et al. Cambrian-1: A Fully Open, Vision-Centric Exploration of Multimodal LLMs. In *NeurIPS*, 2024.

[65] Xingyu Fu, Yushi Hu, Bangzheng Li, et al. BLINK: Multimodal Large Language Models Can See but Not Perceive. In *ECCV*, 2024.

[66] Wufei Ma, Haoyu Chen, Guofeng Zhang, et al. 3DSRBench: A Comprehensive 3D Spatial Reasoning Benchmark. In *ICCV*, 2025.

[67] Dan Hendrycks, Collin Burns, Steven Basart, et al. Measuring Massive Multitask Language Understanding. In *ICLR*, 2021.

[68] Yubo Wang, Xueguang Ma, Ge Zhang, et al. MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark. In *NeurIPS*, 2024.

[69] Peter Clark, Isaac Cowhey, Oren Etzioni, et al. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[70] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a Machine Really Finish Your Sentence? In *ACL*, 2019.

[71] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. In *AAAI*, 2020.

[72] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. In *NAACL-HLT*, 2019.

[73] Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring How Models Mimic Human Falsehoods. In *ACL*, 2022.

[74] NVIDIA. Physical AI autonomous vehicles NuRec dataset. https://huggingface.co/datasets/nvidia/PhysicalAI-Autonomous-Vehicles-NuRec, 2025.

[75] Qi Wu, Janick Martinez Esturo, Ashkan Mirzaei, Nicolas Moenne-Loccoz, and Zan Gojcic. 3DGUT: Enabling distorted cameras and secondary rays in Gaussian splatting. In *CVPR*, 2025.

[76] Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. TransFuser: Imitation with Transformer-Based Sensor Fusion for Autonomous Driving. *IEEE TPAMI*, 45(11):12878–12895, 2023.

[77] Chengran Yuan, Zhanqi Zhang, Jiawei Sun, et al. DRAMA: An efficient end-to-end motion planner for autonomous driving with Mamba. *arXiv preprint arXiv:2408.03601*, 2024.

[78] Bencheng Liao, Shaoyu Chen, Haoran Yin, et al. DiffusionDrive: Truncated diffusion model for end-to-end autonomous driving. In *CVPR*, 2025.

[79] 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.

[80] Yuzhou Huang, Benjin Zhu, Hengtong Lu, et al. MindVLA-U1: VLA beats VA with unified streaming architecture for autonomous driving. *arXiv preprint arXiv:2605.12624*, 2026.

[81] Weicheng Zheng, Yixin Huang, Qiao Sun, Derun Li, and Hang Zhao. DriveMA: Driving vision-language-action models with verifiable meta-actions. *arXiv preprint arXiv:2605.31271*, 2026.

[82] Jinghui Lu, Jiayi Guan, Zhijian Huang, et al. Xiaomi OneVL: One-step latent reasoning and planning with vision-language explanation. *arXiv preprint arXiv:2604.18486*, 2026.

[83] Zihan You, Zehua Fan, Wenqi Lyu, et al. Language2Drive: Do vision-language-action models really follow the instructions in autonomous driving? *arXiv preprint arXiv:2610.16634*, 2026.

[84] Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. In *NeurIPS*, 2019.

[85] Chuan Wen, Jierui Lin, Trevor Darrell, Dinesh Jayaraman, and Yang Gao. Fighting copycat agents in behavioral cloning from observation histories. In *NeurIPS*, 2020.

[86] Alina Kuznetsova, Hassan Rom, Neil Alldrin, et al. The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*, 128(7):1956–1981, 2020.

[87] Kaiyu Yang, Olga Russakovsky, and Jia Deng. SpatialSense: An Adversarially Crowdsourced Benchmark for Spatial Relation Recognition. In *ICCV*, 2019.

[88] Holger Caesar, Varun Bankiti, Alex H. Lang, et al. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020.

[89] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge. In *NAACL-HLT*, 2019.

[90] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense Reasoning about Social Interactions. In *EMNLP-IJCNLP*, 2019.

[91] Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Cosmos QA: Machine Reading Comprehension with Contextual Commonsense Reasoning. *arXiv preprint arXiv:1909.00277*, 2019.

[92] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about Physical Commonsense in Natural Language. *arXiv preprint arXiv:1911.11641*, 2019.

[93] Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. MedMCQA : A Large-scale Multi-Subject Multi-Choice Dataset for Medical domain Question Answering. *arXiv preprint arXiv:2203.14371*, 2022.

[94] Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What Disease does this Patient Have? A Large-scale Open Domain Question Answering Dataset from Medical Exams. *arXiv preprint arXiv:2009.13081*, 2020.

[95] Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. QASC: A Dataset for Question Answering via Sentence Composition. *arXiv preprint arXiv:1910.11473*, 2019.

[96] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. *arXiv preprint arXiv:1809.02789*, 2018.

[97] Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program Induction by Rationale Generation : Learning to Solve and Explain Algebraic Word Problems. *arXiv preprint arXiv:1705.04146*, 2017.

[98] Alisa Liu, Swabha Swayamdipta, Noah A. Smith, and Yejin Choi. WANLI: Worker and AI Collaboration for Natural Language Inference Dataset Creation. *arXiv preprint arXiv:2201.05955*, 2022.

[99] Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, et al. Abductive Commonsense Reasoning. *arXiv preprint arXiv:1908.05739*, 2019.

[100] Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies. *arXiv preprint arXiv:2101.02235*, 2021.

[101] Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as Soft Reasoners over Language. *arXiv preprint arXiv:2002.05867*, 2020.

[102] Holger Caesar, Juraj Kabzan, Kok Seang Tan, et al. NuPlan: A closed-loop ML-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*, 2021.

[103] Mengtian Li, Yu-Xiong Wang, and Deva Ramanan. Towards streaming perception. In *ECCV*, 2020.

[104] David L. Woods, John M. Wyma, E. William Yund, Timothy J. Herron, and Bruce Reed. Factors influencing the latency of simple reaction time. *Frontiers in Human Neuroscience*, 9:131, 2015.

Appendix

A Training data

This section lists the training data of the three stages: the multiple-choice questions of stages 1 and 2 (Tab. 9) and the rows of stage 3 (Tabs. 10 to 12), with how each source is built and what is kept out of training.

Answer readout. Every question of the first two stages is a multiple-choice question with options labelled A, B and so on. The prompt lists the options and ends with 'Answer with only the letter of the correct option.' The answer distribution is the softmax over the logits of the option letters predicted at [ANS], the last prompt position, so one forward pass gives the whole distribution; yes/no questions are two-option questions. Figure 8 shows questions of both stages as the model reads them, and Fig. 3 a stage-3 row.

Question pools. Table 9 lists the sources of stages 1 and 2; stage 2 converts only DriveLM's closed-form questions. The sources carry answer cues: SAT lists the gold first in all 139,632 of its questions, and 16,548 of its 19,101 counting questions (87%) are answered 1; the option shuffling and the tempered answer sampling below counter both. Count options are five consecutive integers that start zero to four below the gold, never below zero, so the gold is not always the middle value. Chart values are spaced by the larger of the gold's precision and 15% of the gold, and a jittered box overlaps the true one at IoU 0.3–0.8. The extra option, an answer that rejects every listed option, goes to text questions with three or more options, except WANLI's, whose three labels cover every case. Only single-image questions are kept. Options are shuffled once with a fixed seed, so questions whose options name other options, such as 'all of the above', are dropped. Within a source, a draw picks a question type with probability proportional to $n^{1/2}$ and then an answer with probability proportional to $n^{3/4}$, where n is the number of questions of that type or with that answer. Images with identical bytes or the same Visual Genome or COCO image never cross the training and development splits. The development splits hold 45,826 questions of SAT, GQA, TallyQA, AI2D, ChartQA and TextVQA and 150 DriveLM questions of each kind; DriveLM is split by scene into 558 training, 69 development and 69 test scenes.

Decontamination. We drop 1,688 duplicate questions, 88 questions with more than one correct option, and 8,171 questions on an image of SAT's evaluation splits [53], BLINK [65], CV-Bench [64] or 3DSRBench [66], matched by identical bytes or the same Visual Genome or COCO image. Images added for OpenImages, SpatialSense and DriveLM questions are also dropped within Hamming distance 10 of an evaluation image under a 63-bit DCT hash, among them 175 DriveLM images of the question kinds and 150 of the box and relation kinds. Text questions that share at least 50% of their 10-word shingles (runs of 10 consecutive words) with the questions of the language benchmarks of Tab. 2 or ARC-Easy [69] are dropped as well.

Driving pools. Table 10 lists the rows of each phase of stage 3, Tab. 11 their prompt and Tab. 12 how each source becomes a row. No training pool holds a frame or clip of any evaluation set; the last row of Tab. 12 lists what each source leaves out. Rollout rows teach recovery: their frames follow AlpaSim rollouts of earlier models in 492 NuRec scenes, 327 from PAI-AV's training split and 165 from its validation split, none among the 916 AlpaSim scenarios. The recovery trajectory of a rollout row returns to the human path in about 3 s, never ahead of the human, and slows behind a lead vehicle. Each instruction scene gives three rows on the same frame: one without an instruction, one with an instruction that the logged future follows and one with an instruction that it does not follow. Each row's target is $q \propto \exp(-d/\tau)$, with $\tau = 1$ m on NAVSIM and WOD-E2E and 0.5 m on PAI-AV. Without an instruction, it spreads over all candidates with d the ADE; with an instruction that the logged future follows, over the candidates that follow it; with one that it does not follow, over the feasible candidates that follow it (on NAVSIM, no collision and inside the drivable area; on PAI-AV, no overlap with an obstacle box from 0.5 to 4 s; on every source, acceleration at most 3.5 m/s^2, also from the current speed, yaw rate at most 0.8 rad/s, and lane changes only into a lane of the same direction), with d the distance to the logged future in the direction the instruction leaves free: along the path for lane and turn instructions, across it for all other kinds. Instructions come from 8 templates for each of 12 kinds, and the last two of each kind are used only in development. To keep Language2Drive's wordings out of training, we wrote the templates ourselves: none comes within a string similarity of 0.65 of any of Language2Drive's 904 instruction strings, and no instruction row comes from a Language2Drive log. A development set of 1,000 rows, 200 each from held-out NAVSIM logs, rollout rows, WOD-E2E, held-out PAI-AV training clips and instruction rows, tracks the loss every 1,000 steps; we keep the last checkpoint. Of its rows, 65 come from 40 of the 916 AlpaSim scenarios.

B Training details

Table 13 lists the settings of every stage; the paragraphs below give the objective of stages 1 and 2 and how the Policy Head grows before RL.

Objective. Stages 1 and 2 train on each question in its stored option order and a shifted one, the gold letter moving with its content. The objective is a paired Brier estimator over draws of the answer distribution, which counts a pair

Stage 1: general QA




```
Which kind of furniture is the cat lying on?
A. chair
B. cabinet
C. stool
D. bench
Answer with only the letter of the correct option.
[ANS] → A
```

gqa_13806917 dev split

Stage 2: driving QA




```
[...] What is the observed status of the object in the red box?
A. Stationary
B. Moving
Answer with only the letter of the correct option.
[ANS] → B
```

drivelm_qa_033697 test split




```
This image is from the ego vehicle's front camera.
What is the target action of the ego vehicle?
A. Turn left
B. Turn right
C. Go straight
D. Stay stationary
Answer with only the letter of the correct option.
[ANS] → D
```

drivelm_qa_018666 test split

Figure 8 **Held-out question samples of stages 1 and 2.** Each prompt is shown beside its image as the model reads it, verbatim apart from the cut marked [...], with the gold option highlighted and its letter as the target at [ANS]; the grey tag gives each question's ID and split.

of draws from the two orders as correct only if both are gold, plus a KL term to the model before the stage (Sec. 3.3). With p, p', y and y' as in Sec. 3.3, the reward is twice the share of pairs that hit minus how often two pairs agree. Its expectation, $2p_y p'_{y'} - \sum_c p_c^2 \sum_{c'} p_{c'}'^2$, is one minus the Brier score of $p \otimes p'$ [61]. The gradient from 32 pairs is unbiased: over two to nine options, the mean of 48 draws has a cosine of at least 0.997 with the exact gradient, and a single draw a median cosine of 0.99 at two options and 0.86 at nine.

Runs before the correction. The ablation runs, in the main text and here, use this objective as implemented before we corrected it, except our supervised model without instruction rows (Tab. 7); the models of the main result tables, the SuperGlance rows of Tab. 17 and its last row were retrained from stage 1 after the correction.

Head refit. Before RL, the Policy Head grows from 5,121 to 5,273 rows, adding one for each candidate built by rule (Sec. C): the old rows are kept, and each new row starts as the row of its nearest old candidate with its bias lowered by 4, which leaves the old logits and the most probable candidate unchanged. Only the new rows are then fit, by L-BFGS on frozen features of NAVSIM and PAI-AV training rows, with an L2 pull of weight 10^{-3} toward their start. Computing p_{SFT} over the RL pool takes another 11.4 H200 GPU hours, and extracting the features for the refit 6.6.

C Candidate trajectories

The candidate set combines k-means centres of NAVSIM's human trajectories (Tab. 14), one stationary trajectory, k-means centres of PAI-AV's high-speed driving and, for RL, 152 candidates built by rule for braking and curves.

Construction. The clustered part runs k-means on the human 4-s trajectories of the 92,339 NAVSIM training scenes, each a vector of 8 positions and headings, and adds one stationary trajectory. Table 14 shows how the coverage grows with K; we use 4,096 centres. NAVSIM's training logs contain few scenes above 13 m/s, so k-means on NAVSIM alone leaves few candidates for the highway speeds of PAI-AV. The 1,024 high-speed candidates are the centres of k-means over the 71,090 PAI-AV training trajectories driven at 13 m/s or faster; their 4-s end points lie 52.5 to 190.5 m ahead. Every candidate is scored once in every training scene (Sec. 3.2).

Braking and curve extension. The set has no plan that brakes to a stop from 9 m/s or faster: at 8 m/s, its shortest

Source	Questions about	Options	Questions	Drawn (%)
Stage 1: 1,158,150 questions, 500,000 drawn				
GQA [52]	Relations, attributes and objects (yes/no); colour, material, category	Yes/no, or the gold and 3 others of its kind	435,825	19.91
flipped	Spatial relations, turned into their opposite	Yes/no; the gold is 'no'	12,158	
SAT [53]	Spatial relations, counts, action consequences, goals, perspective	SAT's own: 2 for relations, 5 for counts	139,632	19.91
TallyQA [54]	Counts	5 consecutive integers around the gold	119,859	16.59
ChartQA [56]	Chart values; yes/no	5 values or 5 years around the gold	22,136	5.31
AI2D [55]	Science diagrams	The dataset's	5,777	3.98
TextVQA [57]	Scene text: yes/no and counts	Yes/no; counts as for TallyQA	2,418	0.66
OpenImages [86]	Which of two drawn boxes shows a named object	The true box and a jittered one or a box on another object	17,966	3.65
directly above	Whether one object is directly above another in the real world	Yes for stacked pairs, such as a cup on a saucer; no for offset or swapped pairs	2,488	1.99
SpatialSense [87]	'Is the *subject predicate* the *object*?'	Yes/no	10,339	1.99
Text, 13 datasets	Commonsense, science, medicine, mathematics, inference and rules, with a blank image	The dataset's: 2 to 8, or yes/no	309,553	19.68
answer checks	'Proposed answer: … Is the proposed answer correct?'	Yes/no; the proposed answer is the gold in half	39,999	4.32
extra option	'None of the other options is correct'	Replaces the gold (half) or joins the options (half)	40,000	1.99
Stage 2: 1,240,040 questions, 192,000 drawn				
DriveLM [3, 88] objects	An object in a red box on the camera that shows it: motion, future state, appearance, relevance to the ego vehicle, type	2 to 5, one answer set per kind; appearance: the gold and 3 descriptions that share no word with it	86,373	30
ego vehicle	Front camera: target action, steering, speed, signals, traffic elements	2 to 5, one answer set per kind	15,680	14
relations	Two key objects: left or right, closer or farther	Yes/no, each with a word-flipped twin	11,352	14
box choice	Which of two drawn boxes shows a key object	The true box and a jittered one	10,973	12
Stage-1 questions	Stage 1 minus directly-above and extra-option rows	As in stage 1	1,115,662	30

Table 9 Question sources of stages 1 and 2. Questions: training questions in the pool; drawn: share of the draws in training. Indented rows are built from the source above, and GQA's flipped questions count toward its share. Text: CSQA [89], SIQA [90], CosmosQA [91], PIQA [92], MedMCQA [93], MedQA [94], QASC [95], OpenBookQA [96], AQuA [97], WANLI [98], αNLI [99], StrategyQA [100] and RuleTaker [101], at most 40,000 questions each.

stop is 11.2 m. The extension appends 152 candidates: 84 that brake from every speed between 5 and 14 m/s and from 16, 18, 20 and 22 m/s, 52 of them to a stop within 4 s, and 68 medium-speed curves and ramps, which the clustered set covers poorly on PAI-AV. Added to the reward-target model with the current frame only (Tab. 17) by fitting only the new rows of its Policy Head, as before RL (Sec. B), the extension keeps navtest PDMS at 90.81, lowers the PAI-AV ADE at 3 s from 0.397 to 0.378 m, and raises the AlpaSim distance per at-fault incident from 0.993 to 1.018 km. The RL rows of the main tables use the extended set, 5,273 candidates. With RL, the extension again lowers the PAI-AV ADE and leaves navtest and the mean scenario score level: in two RL runs from the supervised model before its last phase, one per set, it moves navtest PDMS from 91.65 to 91.60, lowers the PAI-AV ADE at 3 s from 0.403 to 0.383 m and raises the AlpaSim distance per at-fault incident from 1.301 to 1.369 km; the mean scenario score changes by $+0.003$ (95% interval -0.012 to $+0.018$).

D Benchmark protocols

Table 15 summarizes the five benchmarks; the paragraphs below give how we read one plan from our model and how each benchmark runs and scores it. On every benchmark, the front image follows the camera model of nuPlan's CAM_F0 (Tab. 12), re-projected where needed. Unless noted otherwise, other methods' results are as reported in their papers, and TransFuser's by NAVSIM [50].

Plan readout. On every benchmark, our plan is the mean of our 64 most probable candidates, weighted by their probabilities at temperature 0.2; the number was fixed in advance and the temperature set on the PAI-AV validation split. On that split, any number from 32 to all candidates keeps the 3-s and 5-s ADE of our supervised and final models within 0.0009 and 0.0017 m of those at 64, while the most probable candidate alone raises the 3-s ADE by 0.028 and 0.031 m. A temperature of 0.2 gave the lowest 3-s ADE among 1, 0.5, 0.2, 0.1, 0.05 and 0.02 for each of three earlier checkpoints.

Source	Rows	Drawn (%)	Target q or reward r	Rows from
Supervised: 2,191,689 rows, 20,900 steps of 128 rows, 5,121 candidates				
NAVSIM scenes	92,339	12.17	$q \propto \exp\big((\mathrm{PDMS} \cdot \mathrm{DDC} - 0.1\,\mathrm{ADE})/0.1\big)$	One per navtrain scene of 1,083 logs
NAVSIM frames	44,243	5.83	$q \propto \exp(-\mathrm{ADE}/1\,\mathrm{m})$	Other frames of 1,076 of these logs that pass the navtrain filter
PAI-AV logged	1,044,771	36	$q \propto \exp(-\mathrm{ADE}/0.5\,\mathrm{m})$	149,253 training clips, every 2 s
PAI-AV rollouts	46,583	4.5	As above, with the ADE to a recovery trajectory	AlpaSim rollouts of earlier models in 492 scenes
WOD-E2E	406,205	18	$q \propto \exp(-\mathrm{ADE}/1\,\mathrm{m})$	Every 10-Hz frame of 1,991 training segments
Instructions	301,158	10	$q \propto \exp(-d/\tau)$, τ of the source, over the candidates that follow the instruction (see text)	100,386 scenes of the three sources, three rows each
DriveLM	124,378	9.45	Stage-2 objective, KL 0.1 to the stage-2 model	All training questions of stage 2
Other questions	132,012	4.05	As for DriveLM	22 stage-1 sources, among them SAT, GQA and TallyQA
Last phase: 1,846,944 rows, 2,500 steps of 128 rows, 5,121 candidates				
WOD-E2E, rated range	61,460	40	$q \propto \exp\big((\tilde{R} - \mathrm{ADE}/2)/0.5\big)$	Frames 135–165, around the rated frames 147–150, of 1,987 training segments, in place of the WOD-E2E rows
Other rows	1,785,484	60	Their supervised targets	The other supervised rows, each weight $\times\, 0.7317$
RL: 391,322 rows, 2,000 steps of 64 rows, 5,273 candidates				
NAVSIM scenes	92,339	50	$r = \mathrm{PDMS} \cdot \mathrm{DDC}$, minus 10 if NC, DAC or TTC is below 1	The navtrain scenes above
PAI-AV logged	298,504	45	$r = -0.2\,\mathrm{ADE}$	Up to two per training clip
WOD-E2E rated	479	5	$r = \mathrm{RFS}/10$	The rated validation frames

Table 10 **Training pools of stage 3**, by phase. Drawn: share of the rows drawn in the phase; ADE: metres to the logged future over the eight poses at 0.5–4 s; $\tilde{R}$: the RFS with the logged future as the only rated trajectory. None of the phases trains on the 10,949 scenes of NAVSIM's 109 held-out training logs.

`You are driving the ego vehicle; the image is its front camera. Speed` *v* `m/s, longitudinal acceleration` *a* `m/s^2, navigation command:` *c*`. Past positions over the last 1.5 s relative to now (x ahead, y left, metres):` (*x*, *y*), (*x*, *y*), (*x*, *y*)`. Choose the trajectory for the next 4 seconds.`

In one NAVSIM training row: *v* = 10.5, *a* = +0.3, *c* = `go straight`, and the past positions `(-14.6, -0.1), (-10.0, -0.1), (-5.1, -0.0)`.

With past frames: `the image is its front camera` becomes `the images are its front camera 1.5 s, 1.0 s and 0.5 s ago (frames 0-2) and now (frame 3)`, and `frame:` *k* precedes each image, oldest first.

Rows with an instruction: the command is `go straight`, and the prompt ends with `Driving instruction:` and the instruction, such as `Bring the car to a stop and stay stopped.`

Table 11 **Stage-3 prompt**, the same for every driving source. The chat template runs with thinking off, and `[ANS]` is the last position, after its empty thinking block.

NAVSIM. We score navtest with the NAVSIM v1.1 PDM scorer; 12,131 of its 12,146 scenes have all past frames, and the others are read from the current frame only.

WOD-E2E. We report the test split, scored by the official server, since the 479 rated validation frames, the only ones with ratings for every trajectory, provide the RL rewards. Where we compute RFS ourselves, on the rated validation frames and for $\tilde{R}$, we follow the official rater-feedback code. The input is the current frame only, and our 4-s plans are resampled to the benchmark's 4-Hz, 5-s grid, the last second extended at constant velocity.

PAI-AV. The 700 frames are Qwen-Drive-1.0's leakage-free subset: the five timestamps of each of the 140 held-out test clips that it selected. The logged future comes from our processing of the released egomotion and is the reference for every entry we ran in Tab. 6. Our plan's eight poses over 4 s are interpolated to the 10 Hz of the log by a cubic spline that starts at the logged velocity, and extended to 5 s at constant velocity. One plan is our top-64 mean and Qwen-Drive-1.0's first sample; the best of six is the minimum ADE over our six most probable candidates and over its six samples. Qwen-Drive-1.0's paper reports a mean ADE over six samples of 0.42 and 0.47 m at 3 s for its supervised and RL planners.

	NAVSIM	PAI-AV	WOD-E2E
Camera	nuPlan's [102] front camera, CAM_F0, as logged	The 120° f-theta front-wide camera, re-projected into CAM_F0	FRONT, FRONT_LEFT and FRONT_RIGHT, re-projected into CAM_F0 by rotation, each pixel from the camera with the nearest optical axis
Current frame	1920×1080 with CAM_F0's intrinsics and distortion; re-projected frames are saved at JPEG quality 92. The processor resizes the frame to 1024×576, its limit of 589,824 pixels, which gives 576 visual tokens.		
Past frames	1.5, 1.0 and 0.5 s earlier, from the 2-Hz log	The same times, nearest frame within 0.1 s; clips with an even hash, about half	None
	556×313 (at most 174,080 pixels), 544×320 in the processor, 170 visual tokens each. Rows without all three, rollout rows and question rows stay single-frame; 237,246 of the 391,322 RL rows have past frames.		
Sampling	One row per navtrain scene (4 past and 10 future frames, a route), and other frames of the same logs that pass this filter	Every 2 s from 1.9 s after a clip's start to 4.4 s before its end; RL: up to 2 per clip	Every 10-Hz frame; 46 segments held out by hash
Speed and acceleration	Current ego status	Egomotion at about 10 Hz, central differences over ±0.2 s	Last past state, at 4 Hz
Past positions	Logged poses at 2 Hz	Egomotion at -1.5, -1.0 and -0.5 s	Past states at 4 Hz
Input dropout	None	Training rows with an odd token hash: acceleration 0 and past positions at constant speed along the path, so the plan cannot just repeat the past speed changes; evaluation keeps the real values	None
Command	The devkit's: left, straight or right; frames with 'unknown' dropped	From the heading 4 s ahead: left above 20°, right below $-20°$, else straight	The frame's intent: straight, left, right or unknown
Logged future	Poses at 2 Hz up to 4 s	Egomotion at 0.5, 1.0, …, 4.0 s	Every second 4-Hz state up to 4 s
Candidate scores	NAVSIM v1.1 PDM scorer: NC, DAC, EP, TTC, Comf., DDC and PDMS	ADE to the logged future, or to the recovery trajectory in rollout rows	ADE; $\tilde{R}$ in the last phase; for RL, RFS on the 479 rated frames
Excluded	navtest; the 109 held-out logs	Logged and instruction rows: clips outside the training split, the 916 AlpaSim scenes, Language2Drive's AlpaSim scenes and our evaluation clips; rollout rows: only test-split and evaluation clips	Validation and test segments, except the 479 rated frames for RL

Table 12 Driving inputs by source. All sources share the prompt of Tab. 11 and the camera model of CAM_F0; positions are in the current ego frame, speed is $|v|$ and acceleration longitudinal.

AlpaSim. All runs use the 916 scenarios of PAI-AV-NuRec version 26.01, the same simulator configuration and the same scorer, with the ClipGT maps for the off-road check. Other agents replay their logs without reacting, and actors present for under 3 s are dropped; a collision is at fault when it touches the front or a side of the ego vehicle, never when it touches the rear. An incident is a collision or an off-road event; a scenario passes if it has no at-fault collision, off-road event or lateral-corridor exit, and its score is progress along the human path divided by 0.8, capped at 1, or zero if it fails. Mean scores and pass rates are over all 916 scenarios; those whose route fails AlpaSim's sanity check, as it folds back on itself or strays more than 10 m from the logged trajectory, 12 to 14 in our runs and 8 to 12 in Qwen-Drive-1.0's, count as failed, with score zero. Collision and off-road rates are over the scenarios whose route check passes. Scored with the XODR maps instead, Qwen-Drive-1.0's supervised and RL planners reach 0.372 and 0.359 km per at-fault incident, against 0.27 and 0.37 km in its paper, on version 26.02. With a fixed configuration, AlpaSim is deterministic: reruns on other machines, of up to 441 scenarios, matched every metric in every scenario, so each row is one run.

Low-level controller. Every plan passes through our low-level controller, the PDM simulator of Sec. 3.1, which starts a scenario's first plan from the steering angle estimated from the yaw rate; Qwen-Drive-1.0's enters it as the eight of its 50 poses at 0.5, 1.0, …, 4.0 s, and both models read the ego speed and acceleration from AlpaSim's dynamic state. Without the PDM simulator (Tab. 17), AlpaSim's controller tracks the eight poses of our plan directly. Our RL model then drives 0.908 against 1.318 km per at-fault incident and its mean scenario score is 0.106 lower (95% interval 0.077 to 0.134). It leaves the road in 12.6% of the scenarios against 7.6%, while its collision rates change little, so the drop in km per at-fault incident comes mostly from leaving the road. With plans tracked directly and speed differenced from poses, Qwen-Drive-1.0's supervised and RL planners drive 0.306 and 0.246 against 0.461 and 0.383 km per at-fault incident.

Delay. The simulator calls the policy every 100 ms of simulated time and stops its clock while the policy computes;

	Stage 1	Stage 2	Stage 3, supervised	Last phase	RL
Starts from	Qwen3.5-9B, new LoRA	Stage-1 model, merged; new LoRA	Stage-2 model, merged; new LoRA; Policy Head at zero	The supervised model	The last-phase model; Policy Head refit to 5,273 rows
Trainable	80,216,064	80,216,064	80,216,064 + 20,980,737	80,216,064 + 20,980,737	80,216,064 + 21,603,481
LoRA	Rank 32, α 64, dropout 0, on 200 projections: q, k, v and o of the 8 full-attention layers; in_proj_qkv, in_proj_z and out_proj of the 24 Gated DeltaNet layers; gate, up and down of all 32 MLPs. The vision tower, the embeddings and the language-model head stay frozen.				
Objective	Paired Brier, 32 draws per option order	As stage 1	Cross-entropy to q; questions as in stage 2	As stage 3	Exact expected reward
KL term	0.1 to Qwen3.5-9B	0.1 to the stage-1 model	0.1 to the stage-2 model, on question rows	As stage 3	$\beta = 0.3$ to p_{SFT}
Batch	64	64	128	128	64
GPUs $\times$ micro $\times$ accum.	$8 \times 4 \times 2$	$8 \times 4 \times 2$	$8 \times 4 \times 4$	$8 \times 4 \times 4$	$8 \times 4 \times 2$
Steps	7,813	3,000	20,900	2,500	2,000
Rows drawn	500,000	192,000	2,675,200	320,000	128,000
Learning rate	10^{-4}	10^{-4}	10^{-4}	5×10^{-5}	5×10^{-5}
Warm-up	3%	3%	5%	5%	5%
Weight decay	0.01	0.01	0.01	0.01	0.01
Optimizer	AdamW (0.9, 0.999, $\epsilon = 10^{-8}$) with cosine decay after a linear warm-up, gradient norm clipped at 1.0, bfloat16				
Image pixels	65,536–262,144	65,536–262,144	Current frame 65,536–589,824; past frames at most 174,080		
Seed	20260918	20260926	20261022	20261022	20261023
Kept checkpoint	Best on development (the last step)	Best on development, scored every 1,000 steps	Last	Last	Last
GPU hours	31.6 (H100)	11.6 (H100)	145.0 (H200)	16.7 (H200)	7.7 (H200)

Table 13 Training settings. Every stage trains a LoRA of the same shape on the language model, and stage 3 adds the Policy Head. The last phase continues the supervised model of stage 3, and RL the last-phase model, without merging. Batches and draws count questions in stages 1 and 2 and rows in stage 3. Candidate rewards take about 1,600 single-thread CPU process-hours, and rendering the frames of one AlpaSim rollout for the rollout rows takes about 1.1 H200 GPU minutes.

Candidate set	K	Spacing (m)	Best PDMS	Nearest PDMS
k-means 256 + stationary	257	0.98	99.21	77.88
k-means 1,024 + stationary	1,025	0.61	99.66	82.76
k-means 4,096 + stationary	4,097	0.40	99.84	85.69

Table 14 Coverage of NAVSIM scenes by k-means sets of human trajectories. On 10,549 of the 10,949 held-out training scenes, the columns give the mean distance between neighbouring candidates and the mean PDMS of the best candidate and of the candidate nearest to the human trajectory. More centres mainly lift the nearest candidate; the best one is near the PDMS ceiling at every size. Shading deepens from worst to best in each PDMS column.

the policy returns its full plan. Following streaming perception [103], we charge each model's computation time as delay. For Tab. 3 (top), we charge our 84 ms and Qwen-Drive-1.0's 518 ms as measured and its 1,213 ms with reasoning as 1.2 s; the simulator rounds a delay up to the next 100 ms, so these plans reach AlpaSim's controller 100, 600 and 1,200 ms after their call. A late plan keeps the timestamps of its call: the PDM simulator runs it from the ego pose, speed and acceleration at the call and from the steering angle that its run of the previous plan reaches at that time, and AlpaSim's controller anchors it at the ego pose of its first timestamp and tracks it from the current time on, with no prediction of the ego state over the delay. Until the first plan arrives, AlpaSim's controller follows the logged path. Over 2,000 resamples of the scenarios, the 95% intervals of km per at-fault incident are 1.06 to 1.51 and 0.240 to 0.282 for our RL model at 84 ms and 1.2 s and 0.289 to 0.351 and 0.148 to 0.170 for Qwen-Drive-1.0 at 518 ms and 1.2 s; without delay, they are 1.01 to 1.44 and 1.11 to 1.60 for our supervised and RL models and 0.410 to 0.521 and 0.342 to 0.436 for Qwen-Drive-1.0's supervised and RL planners.

Fixed lags. In Fig. 6 (bottom), every plan arrives a fixed lag after its call, whatever its computation time, and each planner absorbs the lag up to a limit. With lags of 0, 100, 300, 600, 1,200, 2,000 and 5,000 ms, our RL model drives 1.318, 1.248, 1.224, 1.061, 0.259, 0.212 and 0.192 km per at-fault incident and passes 73.5, 72.2, 70.9, 66.0, 16.2, 8.4 and 41.5% of the scenarios; at 0, 300, 600, 1,200, 2,000 and 5,000 ms, Qwen-Drive-1.0 without reasoning drives 0.470, 0.502, 0.317, 0.165, 0.150 and 0.158 km and passes 60.0, 59.9, 36.6, 9.8, 3.2 and 33.5% (Fig. 6, bottom). Against no delay, Qwen-Drive-1.0 drives 1.07 times as far at 300 ms (95% interval 0.99 to 1.17), 0.68 times at 600 ms (0.60 to 0.76), 0.35 times at 1.2 s (0.31 to 0.39), 0.32 times at 2 s

Benchmark	What it is	Loop	Split and size	Metrics	Horizon
NAVSIM [50] v1.1	Scenes of nuPlan [102] logs; the PDM scorer simulates each plan	Open	navtest: 12,146 scenes from 136 logs	PDMS: NC × DAC × a weighted sum of EP, TTC and Comf.	4 s at 2 Hz
WOD-E2E [59]	Long-tail scenes; RFS scores a plan against rated trajectories	Open	Test split, scored by the official server; the 479 rated validation frames provide RL rewards	RFS; 3- and 5-s ADE	5 s at 4 Hz
PAI-AV [58]	Driving clips with released egomotion	Open	Qwen-Drive-1.0's leakage-free subset: 700 frames, five in each of 140 held-out test clips	3- and 5-s ADE of one plan and of the best of six	5 s at 10 Hz
AlpaSim [13]	Simulator that renders the camera along the car's own path in PAI-AV-NuRec [74, 75] v26.01 scenes	Closed	916 scenarios; one whose route check fails counts as failed, with score zero	Collision and off-road rates, progress, km per at-fault incident, pass rate and mean scenario score	Whole scenario, a call every 100 ms
Language2Drive [83]	Two instructions per scene; checks whether each plan follows its own (NAVSIM track)	Open	38 instruction pairs in four wordings each	SemSISR, SemCFPS: single instructions and pairs followed; SCS, PCS: graded scores	4 s

Table 15 The five benchmarks. Four score each plan once (open loop); in AlpaSim, the car drives its plans (closed loop). Where a benchmark scores 5 s, our 4-s plans are extended at constant speed.

(0.29 to 0.36) and 0.34 times at 5 s (0.30 to 0.38); our model drives 0.93 times as far at 300 ms (0.84 to 1.02), 0.80 times at 600 ms (0.72 to 0.90), 0.20 times at 1.2 s (0.17 to 0.23), 0.16 times at 2 s (0.14 to 0.19) and 0.15 times at 5 s (0.12 to 0.17). Past the limit, successive plans stop agreeing and the car oscillates: the mean distance between plans made 0.1 s apart is 0.06 and 0.07 m for Qwen-Drive-1.0 at 0 and 300 ms and 0.65 m at 600 ms; for our model it is 0.07, 0.07 and 0.15 m at 0, 300 and 600 ms and 0.47 m at 1.2 s. At 1.2 s, our model drives 0.82 times as far as Qwen-Drive-1.0 at its 518 ms (0.74 to 0.90). At equal lags, the lead narrows beyond 600 ms: our model drives 2.81 times as far as Qwen-Drive-1.0 when neither is delayed (2.38 to 3.37), 1.57 times at 1.2 s (1.45 to 1.71), 1.42 times at 2 s (1.33 to 1.52) and 1.21 times at 5 s (1.12 to 1.31). A 5-s lag outlasts a 4-s plan: every plan arrives after its last pose, and AlpaSim's controller tracks that pose. Both cars then fall behind the log, each covering a median of 75% of the human's distance over the same span (99% and 98% at 2 s), and the replayed vehicle behind hits ours in 303 of the 916 scenarios and Qwen-Drive-1.0's in 278. A hit from behind is not at fault and ends the scenario: 302 of our 380 passes and 278 of Qwen-Drive-1.0's 307 are scenarios that end this way, so both pass rates rise from 2 s to 5 s while km per at-fault incident moves by less than 10%. Rollout rows still help under delay: at 100 ms, our supervised model before its last phase drives 1.395 km per at-fault incident, and 1.026 km when trained without them, a ratio of 0.74 (95% interval 0.62 to 0.87), against 0.64 (0.54 to 0.77) without delay.

E Latency

This section gives the protocol behind the GPU times per plan in the main text, and Tab. 16 adds the other planners that we timed.

Method	Sequential passes	GPU (ms)	End to end (ms)
SuperGlance	1	83.8	145.5
ReCogDrive [5]	6	198.6	296.5
Qwen-Drive-1.0 [1]	11	518.3	1,014.1
+ reasoning	28	1,213.2	1,710.1
AutoVLA [2]	45	1,529.3	1,900.3
Text waypoints	∼155	5,557.7	5,622.4

Table 16 Time per plan. GPU time grows with the number of sequential passes. Times are medians of 100 calls, two on each of 50 navtest scenes, on one H200 at batch size one in eager PyTorch; end to end is the median of each call's total, which adds input processing (image decoding, resizing and tokenization) to the GPU time. With reasoning and for text waypoints, the number of passes varies by call, and the table gives its median. Shading deepens from the slowest to the fastest time in both time columns.

Each model runs in its own process on one H200 at batch size one, in eager bfloat16 PyTorch without compilation or CUDA graphs, on a GPU with no other job. On the same 50 navtest scenes, it makes at least 40 s of untimed calls and five warm-up calls, then 100 timed calls, two on each scene; its GPU time is the median over these calls (Tab. 16). SuperGlance runs with its history frames and LoRA weights merged; reusing the vision features of past frames and skipping per-call bookkeeping brings it to 73 ms, with identical outputs on these scenes; we report the plain time, as the other planners run without such changes. ReCogDrive runs its diffusion planner in fp32 with five DDIM steps, Qwen-Drive-

Model	Collision (%)	Off-road (%)	Progress (%)	km / incident	Mean score	Pass (%)
Qwen-Drive-1.0 + reasoning	32.8 / 14.4	15.4	71.1	0.284 / 0.461	0.516	58.7
Qwen-Drive-1.0 + reasoning, RL	43.6 / 15.6	13.1	57.3	0.194 / 0.383	0.391	53.6
Imitation, current frame	15.4 / 9.1	9.8	84.5	0.684 / 0.914	0.603	64.4
Imitation, history frames	10.3 / 5.1	9.8	87.4	0.888 / 1.199	0.682	71.0
Reward target, current frame	15.1 / 8.1	9.3	84.5	0.708 / 0.993	0.608	65.3
Reward target, current frame, extended set	15.3 / 8.1	8.9	84.5	0.714 / 1.018	0.611	65.6
Reward target, history frames, last phase (SuperGlance)	11.2 / 6.0	8.8	87.9	0.886 / 1.199	0.700	72.1
Reward target, current frame + sampled RL	13.5 / 9.0	7.2	87.9	0.861 / 1.103	0.649	66.9
Reward target, current frame + exact RL, normalized	13.4 / 8.3	7.5	86.7	0.842 / 1.113	0.647	67.1
Reward target, current frame + exact RL	14.0 / 8.3	7.9	85.4	0.796 / 1.074	0.628	66.6
Reward target, current frame + exact RL, $\beta = 0.1$	14.7 / 9.4	8.2	85.5	0.751 / 0.978	0.608	64.1
Imitation, history frames + exact RL	11.3 / 5.2	5.7	87.9	1.060 / 1.655	0.727	75.9
Reward target, history frames + exact RL, original set	9.0 / 4.8	8.8	87.4	0.992 / 1.301	0.701	72.2
Reward target, history frames, last phase + exact RL (SuperGlance)	10.6 / 5.9	7.6	88.6	0.975 / 1.318	0.717	73.5
no PDM simulator	10.1 / 5.2	12.6	83.3	0.713 / 0.908	0.612	63.3

Table 17 AlpaSim results by training recipe. All rows run the 916 scenarios without delay, with the same simulator configuration and scorer, and our rows read the weighted top-64 mean at each call. Blocks: Qwen-Drive-1.0's planners, our supervised models and our RL models; the indented row runs the model above it without the PDM simulator. Collision and km per incident: all / at-fault. Shading deepens from the worst to the best value in each column. Last phase: the last supervised phase of stage 3, with WOD-E2E rows around the rated frames (Tab. 10); the other supervised rows stop before it.

1.0 its planning expert with ten Euler steps and sdpa attention, faster here than its default FlashAttention-2, and AutoVLA its public NAVSIM checkpoint with greedy decoding, which produced 45 tokens in every scene. The text-waypoint row measures only the cost of writing the plan as text: it uses the same merged model and inputs as SuperGlance and decodes greedily as many tokens as the human trajectory takes in ReCogDrive's text format, 154.5 on the median. Timed in a separate run of 50 calls, the model trained to write this text (Tab. 8) takes 5,382 ms to write its own plans, a mean of 150.5 tokens, 66 times the 81.8 ms of the regression model in that run. Input processing takes 61.7 ms for our four frames and 492.7 ms for Qwen-Drive-1.0's official processor. The dashed line in Fig. 1 marks 231 ms, the mean simple visual reaction time of 1,469 adults [104].

F Additional results

This section gives the evidence behind the main results: how our driving in AlpaSim differs from Qwen-Drive-1.0's (Fig. 9), written replies against the letter read, the drop on TruthfulQA, the parts of the recipe in closed loop, the RL step, frozen features, the scenes of Fig. 10 and Language2Drive. Closed loop separates what navtest does not: without history frames, rollout rows or PAI-AV rows, navtest moves by under half a point while km per at-fault incident in AlpaSim drops (Tab. 19).

Decoded replies. Table 18 checks the letter read against written replies to the same questions. Replies come from stock vLLM, greedy, at the letter read's image limits, with its question and option text and without its instruction line, at a cap of 4,096 tokens with thinking off and 8,192 with thinking on. We score the last option a reply states; unparsable and cut replies are wrong, and 1,298 of the 17,049 replies with thinking on, which include one for each option order on DriveLM and 3DSRBench, reach the cap. Under the letter read's own prompt, Qwen3.5-9B mostly writes one letter and lands within 0.60 points of the letter read on nine benchmarks; on MMLU and MMLU-Pro it scores 1.10 and 16.50 points above it. The gain of writing comes from replies that reason first: with thinking off, replies that state the answer within 80 characters lie within -0.72 to $+1.38$ points of the letter read, while those that reason first lie +27.16 points above it on MMLU-Pro, +13.17 on WinoGrande and +12.43 on MMLU. On the questions with no cut reply, the final model's letter read differs from Qwen3.5-9B's replies with thinking on by -0.06 on CV-Bench, -1.54 on BLINK, -3.04 on 3DSRBench and $+14.81$ on DriveLM. Decoded, the final model stays below Qwen3.5-9B's replies with thinking off on five of the seven language benchmarks, all but HellaSwag and BoolQ, and mostly writes a label, alone or with its option text, averaging 3 to 4 tokens on six of them.

TruthfulQA. TruthfulQA is the one benchmark on which our letter ends below Qwen3.5-9B's (Tab. 2), mostly through stage 1. The loss falls on questions whose correct answer rejects the premise or declines to answer, such as 'I have no comment': on these, accuracy drops from 75.8% to 61.2% after stage 1 as the model picks a definite claim, while the other questions hold. In TruthfulQA's own phrasings, such as 'It depends', such answers are almost never gold in our stage-1 questions; the extra option of Tab. 9 only says that no listed option is correct. Qwen-Drive-1.0 answers these questions as well as Qwen3.5-9B does.

One change at a time. Table 19 changes one thing at a time in the supervised model before its last phase. Without history frames, the scenarios with an at-fault collision double, from 37 to 73, and front hits on stopped vehicles rise from 10 to



Figure 9 **Closed loop in AlpaSim, without delay.** The plans of Qwen-Drive-1.0's RL planner lean left of the human trajectory, by a median of about 0.4 m at 2–3 s ahead, while ours stay centred. (a) Lateral offset of the car and of its plan from the human trajectory, over the calls in the 916 scenarios made while the car moves and before the first incident: lines give the median, bands the middle 50%. (b, c) Front-camera frames with the plan returned at each frame, and a bird's-eye view at the moment of contact: Qwen-Drive-1.0's plans bend over the centre line into an oncoming car (b) and, at a red light, through the car standing ahead, while ours waits (c).

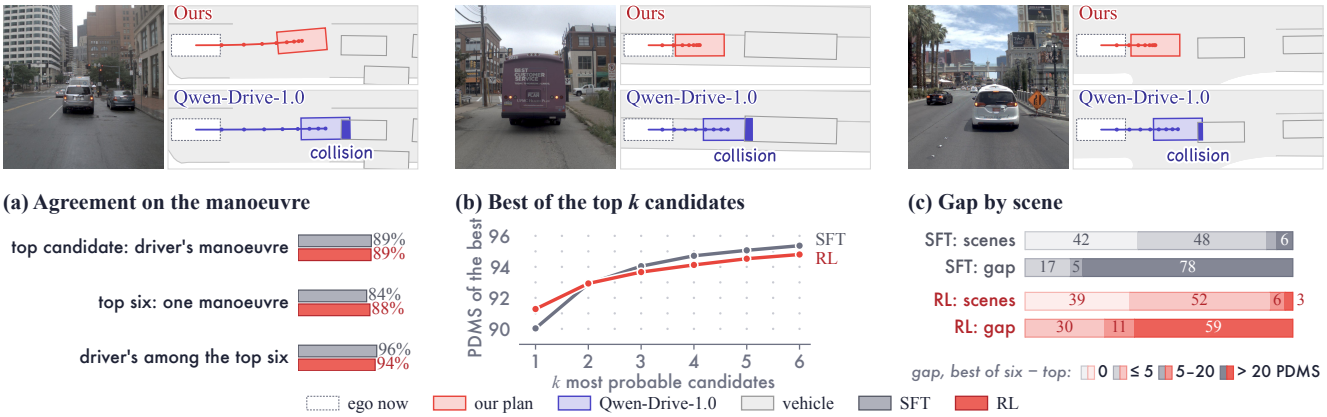


Figure 10 **Manoeuvre and choice on NAVSIM navtest.** The most probable candidate takes the driver's manoeuvre in 89% of the scenes; RL narrows its gap to the best of the six. Top: front camera and NAVSIM map for three of 20 scenes where Qwen-Drive-1.0's plan [1] runs into the vehicle ahead while our final RL model's stays behind it, within 1 m of the human's at 4 s; both plans as the NAVSIM scorer simulates them, in 0.5-s steps, with all vehicles at 4 s. Bottom, over all 12,146 scenes, for our supervised (SFT) and final RL models: (a) manoeuvre agreement of the most probable candidates; (b) PDMS of the best of the k most probable; (c) share of the scenes and of the total gap between the best of the six and the most probable one, by gap size. Manoeuvres, Qwen-Drive-1.0 runs and scene selection: Sec. F.

44, while navtest moves by under half a point. Behaviour cloning on histories is reported to do the reverse, better offline and worse in closed loop, by copying past actions [84, 85]. Rollout rows, 4.5% of the rows drawn, act the same way: without them at-fault scenarios rise to 88, and front hits on moving vehicles from 18 to 59, while navtest does not move. An imitation model with history scores 8.8 PDMS below a reward model without history and drives better: its mean scenario score is 0.075 higher (95% interval 0.047 to 0.101). In mean scenario score, every other variant, all with history frames, stays above the one without them, except the one without PAI-AV rows. That model drives more slowly: progress falls from 88.1% to 64.1%, and vehicles behind hit it in 33.8% of scenarios against 4.7%, so more pass at a lower score. Read as in Tab. 2, DriveLM and 3DSRBench fall from 79.07 and 57.65 to 75.87 and 53.29 without stages 1 and 2, and to 43.33 and 44.85 without question data. Without question data, all eleven benchmarks fall below Qwen3.5-9B's letter in Tab. 2; the other nine score 84.00 (CV-Bench), 75.64 (BLINK), 72.86 (MMLU), 45.84 (MMLU-Pro), 91.72

		Qwen3.5-9B				Final model	
Benchmark	Q	Letter	Decoded	Thinking	Capped (%)	Letter	Decoded
CV-Bench	1,320	87.68	87.05	85.36	6.36	88.55	88.61
BLINK	509	79.37	76.62	75.44	10.41	79.37	79.17
3DSRBench	1,000	43.50	58.20	55.00	9.17	58.10	57.60
DriveLM	1,000	61.30	61.50	54.70	13.05	86.90	86.20
MMLU	1,800	76.78	85.50	83.11	6.89	78.94	78.61
MMLU-Pro	1,200	52.17	76.25	70.83	20.42	58.08	56.67
ARC-C	1,172	93.77	96.16	96.67	0.68	95.31	95.05
HellaSwag	1,800	88.94	89.11	87.89	3.50	89.61	89.39
WinoGrande	1,267	72.53	85.64	87.45	3.79	76.40	76.48
BoolQ	1,800	88.61	88.39	90.50	1.11	89.56	89.44
TruthfulQA	817	74.42	76.25	76.62	5.02	65.61	64.50

Table 18 Letter and decoded replies. Each row scores a fixed subset of a benchmark by the option letter and by written replies, and shading deepens along the row from lowest to highest accuracy. Q: questions; Decoded, Thinking: written replies, thinking off and on; Capped: share of thinking replies at the token cap. Table 2 gives Qwen3.5-9B's decoded replies on full benchmarks.

Change	PDMS↑	ADE↓	km↑	Pass↑	Score change (95% interval)
No change	91.30	0.400	1.486	73.4	–
No history frames	90.83	0.397	0.993	65.3	−0.104 (−0.134, −0.076)
No rollout rows	91.35	0.393	0.958	71.3	−0.027 (−0.058, +0.002)
Imitation target	82.03	0.395	1.199	71.0	−0.030 (−0.052, −0.010)
Linear regression	87.82	0.320	1.360	70.3	−0.026 (−0.055, +0.004)
No PDM simulator	–	–	0.929	64.3	−0.090 (−0.114, −0.064)
No WOD-E2E rows	91.48	0.389	1.470	73.6	+0.004 (−0.018, +0.025)
No PAI-AV rows	90.95	1.048	0.826	75.2	−0.115 (−0.148, −0.081)
No stages 1 and 2	91.48	0.393	1.425	74.9	+0.014 (−0.010, +0.038)
No question data	91.30	0.389	1.361	73.4	+0.001 (−0.024, +0.026)

Table 19 One change at a time. Each row changes one thing in the supervised model before its last phase, read at the top-64 mean. Columns: navtest PDMS, PAI-AV 3-s ADE (m) and, in AlpaSim without delay, km per at-fault incident, share passed (%) and the paired change in the mean scenario score with its 95% interval; –: not applicable. Navtest PDMS moves by under a point except with the imitation target or linear regression, while km per at-fault incident falls most without history frames, rollout rows, PAI-AV rows or the PDM simulator. Shading deepens from the worst to the best value in the PDMS, ADE, km and pass columns.

(ARC-C), 81.52 (HellaSwag), 69.61 (WinoGrande), 88.01 (BoolQ) and 64.75 (TruthfulQA). On WOD-E2E and PAI-AV, writing the plan as text is level with the linear regression of Tab. 8: its WOD-E2E validation RFS is 0.001 lower (95% interval −0.161 to +0.154) and, with both plans interpolated linearly, its PAI-AV ADE is 0.010 m higher at 3 s (−0.005 to +0.024) and 0.062 m higher at 5 s (−0.002 to +0.126).

Training recipes in closed loop. Table 17 lists the closed-loop results by training recipe with all metrics of our scorer and the mean scenario score. Before the last phase, history frames cut the scenarios with an at-fault collision from 82 to 46 with imitation targets and from 73 to 37 with reward targets (Tab. 19). The three history frames change navtest PDMS by under a point, from 81.31 to 82.03 with imitation and from 90.83 to 91.30 with reward targets, but matter in closed loop. With imitation targets, they raise km per at-fault incident from 0.914 to 1.199 and the mean scenario score by 0.08 (95% interval 0.05 to 0.11); 106 scenarios pass only with them and 46 only without. Without history frames, the reward-target model still drives 0.993 km per at-fault incident, over twice the 0.461 km of Qwen-Drive-1.0's supervised planner. RL from the imitation model passes the most scenarios, with 1.655 km per at-fault incident and the lowest off-road rate.

	navtest				AlpaSim
RL step	NC↑	TTC↑	EP↑	PDMS↑	km↑
From the reward target, current frame only					
None	98.48	95.03	85.93	90.83	0.993
Sampled, normalized	97.87	93.45	89.12	91.30	1.103
Exact, normalized	98.93	96.50	85.55	91.60	1.113
Exact	99.11	97.16	83.68	91.16	1.074
Exact, $\beta = 0.1$	99.21	97.33	82.53	90.74	0.978
From the imitation target					
Exact	99.35	97.92	79.44	89.79	1.655

Table 20 RL step. Sampling with a normalized advantage trades NC and TTC for progress; the exact expectation raises both. Scores are on navtest and in closed loop on the 916 AlpaSim scenarios; km: km per at-fault incident. Shading compares the rows of the first block, from the worst to the best value in each column. RL runs use 2,000 steps and $\beta = 0.3$ unless noted; "sampled" draws 2×32 candidates per scene, and "normalized" divides the advantage by its standard deviation in each scene, as in GRPO [31].

RL step. We compare three ways to optimize Eq. (3) from the same supervised model (Tab. 20). In each scene, the exact gradient with respect to candidate k's logit is $p_\theta(k \mid x)\left(r(x,k) - \sum_j p_\theta(j \mid x)\, r(x,j)\right)$ plus the KL gradient, and the bracketed advantage is not normalized per scene. Sampling with a normalized advantage, as in GRPO, raises progress most but lowers NC and TTC below the supervised model. The exact expectation raises NC and TTC: it penalizes every candidate that collides or comes too close at every step, whereas a sampled estimate sees the penalty only when it draws that candidate. Normalizing the exact advantage per scene gives the highest PDMS, 91.60, through more progress; we keep the unnormalized advantage, which gives the fewest navtest collisions and time-to-collision violations of the three, by a rule fixed in advance on held-out training scenes. In closed loop, the three drive about equally far per at-fault incident. A smaller KL weight, $\beta = 0.1$, lowers PDMS and does not help in closed loop: over the 916 scenarios its mean score is 0.020 lower than with $\beta = 0.3$ (95% interval 0.000 to 0.040), and 34 scenarios pass only with $\beta = 0.1$ against 57 only with $\beta = 0.3$. Started from the imitation model, the same RL step raises navtest PDMS from 82.03 to 89.79 and the distance per at-fault incident from 1.199 to 1.655 km, so the step also works when the supervised targets carry no reward.

Frozen features. To see what stages 1 and 2 give the hidden state for planning, we freeze the VLM and train only the Policy Head, on the hidden states h of Qwen3.5-9B or of the stage-2 model, with our supervised model's prompt, history frames and 5,121 candidates. Full-batch L-BFGS fits it from zero to the NAVSIM target of Eq. (2) on the 92,339 navtrain scenes alone, with an L2 penalty on W. The penalty weight, a power of ten chosen by the PDMS of the most probable candidate on the 10,949 held-out scenes (Tab. 10), is 10^{-4} for Qwen3.5-9B and 10^{-5} for the stage-2 model. The head reaches 79.44 PDMS on Qwen3.5-9B and 82.17 on the stage-2 model, against 91.30 when stage 3 trains the head and the VLM together on all its rows. On frozen features, the stage-2 model thus scores 2.73 above Qwen3.5-9B and 9.13 below the jointly trained model, with 95% intervals of 1.83 to 3.59 and 8.10 to 10.27 over 2,000 resamples of navtest's 136 logs. End to end, planning needs no stages 1 and 2 (Tab. 19).

Manoeuvres and scenes. In Fig. 10, a manoeuvre is left, straight, right or stop, from the 4-s heading change, lateral offset and path length. For the top row, we run Qwen-Drive-1.0's released RL planner with reasoning through its own script, one sample per scene, on the three cameras at the last four 2-Hz frames, the ego state and the command; its 10-Hz history is interpolated from NAVSIM's 2-Hz poses, and its plan is resampled to 0.5-s steps with its own code and scored like ours. We run it on 998 navtest scenes with three cameras, those in which an earlier model of ours ends within 1.5 m of the driver at 4 s; our final plan does so in 785 (turns, lane changes, pull-aways, stops and following), and Qwen-Drive-1.0's plan hits a vehicle in 30. In 20 of these, ours scores at least 0.9 PDMS and ends within 1 m of the driver while its plan hits a vehicle 3.1–4.0 s in (19 following or stopping, one lane change); the top row shows three.

Instructions. Table 7 gives all scores on the NAVSIM track of Language2Drive [83], for our final model and for the predictions of the five VLAs its paper reports. Our supervised pools hold 301k instruction rows in its style, written for 100k training scenes of NAVSIM, PAI-AV and WOD-E2E. Without the instruction in the prompt, our model follows no pair. With the earlier scorer of the benchmark paper, which reproduces its five rows, our model scores 26.3, 0.0, 27.6 and 23.6 on SemSISR, SemCFPS, SCS and PCS, and the best of the five 8.6, 0.0, 11.2 and 2.2.

G Limitations

SuperGlance drives only weighted means of its 5,273 candidates, so a manoeuvre rare in human driving must be built by rule and trained again: no clustered candidate brakes to a stop from 9 m/s or faster (Sec. C). In AlpaSim without delay, 10 of the final model's 53 at-fault collisions are with a car stopped in or partly in its lane, though a candidate that stops or goes around exists in each. Decoded, the final model trails Qwen3.5-9B's replies on five of the seven language benchmarks in Tab. 18, all but HellaSwag and BoolQ.