PROACTIVEDRIVE: FUTURE-AWARE ADAPTIVE THINKING FOR VISION-LANGUAGE-ACTION DRIVING

Anonymous authors
Paper under double-blind review

ABSTRACT

Anticipating near-future scene evolution has shown strong potential for autonomous driving by enabling policies to foresee how their surroundings may change before acting. However, bringing this capability into vision-language-action (VLA) driving remains challenging, as future prediction and language reasoning can both increase inference cost in closed-loop deployment. We present ProActiveDrive, an adaptive-thinking framework for efficient VLA autonomous driving. Built on a SimLingo-style language-command policy, ProActiveDrive-combines lightweight future-aware representation learning with selective reasoning at inference time. Instead of relying on dense generative rollouts, the model uses a compact feature-level future predictor to provide anticipatory context to the driving policy. At the same time, adaptive thinking avoids unnecessary language generation by activating explicit reasoning only when it is useful. On Bench2Drive-220, ProActiveDriveimproves over the SimLingo language-command baseline, increasing the driving score from 86.07 to 88.20 and the success rate from 67.27% to 71.30%. Ablation studies show that adaptive reasoning improves inference efficiency and that compact near-future prediction is more effective than heavier multi-horizon prediction in closed-loop driving. Evaluation on Fail2Drive further highlights the remaining challenge of out-of-distribution generalization. Our results suggest that combining efficient reasoning with lightweight future-aware prediction can improve both efficiency and decision quality in embodied VLA systems.

1 INTRODUCTION

Closed-loop autonomous driving remains a central challenge for learning-based planning. Unlike open-loop prediction, where a model is evaluated against fixed expert trajectories, closed-loop driving requires a policy to continuously interact with a dynamic environment, where each action affects future observations and can compound errors over time. As a result, closed-loop benchmarks provide a more faithful measure of whether a learned policy can actually drive, rather than merely imitate demonstrations offline (Dosovitskiy et al., 2017; Chitta et al., 2023; Jaeger et al., 2023; Jia et al., 2024; Zimmerlin et al., 2024; Liao et al., 2025).

Recent vision-language-action (VLA) models have emerged as a promising paradigm for autonomous driving by unifying visual perception, language commands, reasoning, and action prediction within a single policy (Renz et al., 2025; Fu et al., 2025; Kong et al., 2025; Zhou et al., 2025; Ling et al., 2026; Jia et al., 2026). SimLingo, for example, combines closed-loop driving with language understanding and language-action alignment through commentary-style reasoning, action-dreaming supervision, and a disentangled waypoint representation (Renz et al., 2025). However, many VLA agents invoke explicit language reasoning at every timestep, regardless of scene complexity. While reasoning can improve decision-making in challenging situations, applying it uniformly may introduce unnecessary computation and latency during closed-loop deployment.

A complementary line of work equips driving policies with anticipatory capabilities through world models and future prediction. Methods such as GAIA-1, DrivingWorld, DeepSight, IDOL, GenieDrive, and *See Tomorrow, Act Today* explore visual imagination, latent-state forecasting, physics-

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

aware generation, and inverse-dynamics-guided prediction to provide future context for planning (Hu et al., 2023a; 2024; Zhang et al., 2026c;b; Yang et al., 2026; Zhang et al., 2026a). However, many of these approaches rely on large generative architectures, dense rollouts, or multi-step simulation, increasing training and inference costs. For example, GAIA-1 employs a 9B-parameter generative driving world model, illustrating the computational scale required by high-fidelity future generation (Hu et al., 2023a). Furthermore, generated futures can accumulate forecasting errors or violate temporal consistency, complicating their direct use for closed-loop control.

These observations suggest a broader challenge: how should a driving agent allocate computational resources between reasoning and anticipation? Existing systems often treat both as mandatory operations applied uniformly across timesteps. Yet many driving situations are routine and require neither elaborate reasoning nor extensive future simulation, whereas intersections, pedestrians, blocked lanes, and complex priority decisions may benefit substantially from both. This raises a fundamental question: can reasoning and future prediction be deployed adaptively according to scene complexity while preserving strong closed-loop performance?

Our motivation is inspired by recent advances in adaptive reasoning for large language models. In mathematical and symbolic reasoning, chain-of-thought, self-consistency, Tree-of-Thoughts, and adaptive-thinking methods demonstrate that explicit reasoning can improve challenging problems while motivating more selective allocation of test-time computation (Wei et al., 2022; Wang et al., 2023; Yao et al., 2023; Chen & Li, 2024; Li et al., 2026a). We argue that a similar principle applies to autonomous driving: easy decisions should be resolved efficiently, while difficult situations should receive additional reasoning and anticipatory computation.

To investigate this hypothesis, we propose ProActiveDrive, an adaptive-thinking framework for efficient VLA autonomous driving. Our central idea is to view language reasoning and future prediction as adaptive computational resources rather than always-on components. Built on a SimLingo-style language-command policy, ProActiveDrive combines two complementary mechanisms. First, adaptive thinking selectively activates explicit reasoning only when additional deliberation is expected to be beneficial. Second, a lightweight feature-level future predictor provides short-term anticipatory context without requiring dense generative rollouts. Together, these components enable the policy to allocate computation according to driving complexity while maintaining efficient closed-loop inference.

We evaluate ProActiveDrive in closed loop on Bench2Drive-220 and further assess generalization on Fail2Drive (Jia et al., 2024; Gerstenecker et al., 2026). Experiments reveal two key findings. First, selective reasoning achieves better driving performance than always-on reasoning while improving inference efficiency. Second, compact single-step future prediction is more effective for closed-loop driving than heavier multi-horizon forecasting strategies. Overall, ProActiveDrive improves driving score from 86.08 to 88.20 and success rate from 67.27% to 71.30% relative to the SimLingo baseline. Evaluation on Fail2Drive further validates the effectiveness of our approach, which remains highly competitive in this challenging benchmark, suggesting that adaptive reasoning and lightweight anticipation improve the robustness of closed-loop driving.

Our contributions are summarized as follows:

- We introduce ProActiveDrive, an adaptive-computation framework for vision-language-action autonomous driving that selectively activates explicit reasoning according to scene complexity.
- We show that lightweight feature-level anticipation can provide useful future context without the cost of dense generative rollouts.
- We demonstrate through closed-loop experiments that adaptive reasoning and lightweight anticipation improve both driving performance and computational efficiency.
- We provide extensive evaluation on Bench2Drive-220 and Fail2Drive, including ablations and out-of-distribution generalization analysis.

2 Related Work

Vision-language-action driving. Recent end-to-end driving systems increasingly integrate visual perception, language understanding, reasoning, and action prediction within a unified policy. Vision-language-action (VLA) approaches leverage visual observations together with language commands,

scene descriptions, or driving-oriented reasoning to improve decision making and trajectory generation (Renz et al., 2025; Fu et al., 2025; Kong et al., 2025; Zhou et al., 2025; 2026). SimLingo (Renz et al., 2025) is a representative closed-loop VLA framework that combines vision-language understanding with language-action alignment through commentary supervision, action dreaming, and disentangled waypoint prediction. Collectively, these works demonstrate the benefits of language-conditioned driving, but largely focus on reasoning over the current observation, while future-aware representations are typically explored in a separate line of research.

Adaptive reasoning and computation. A growing body of work studies how neural systems can allocate computation according to input difficulty instead of applying a uniform inference budget to all examples. Early-exit and confidence-based mechanisms accelerate inference by terminating computation once predictions become sufficiently reliable (Schuster et al., 2022; Jazbec et al., 2024). More recent reasoning models extend this principle to test-time deliberation, including adaptive-thinking methods, dual-process architectures, and difficulty-aware reasoning strategies that selectively invoke additional computation only when beneficial (Snell et al., 2025; Li et al., 2026a; Cheng et al., 2025; Zhang et al., 2025; Liu et al., 2026). Although these methods have primarily been studied in language and reasoning tasks, they offer a broader perspective on adaptive computation. We draw inspiration from this literature and investigate whether reasoning can be allocated selectively within a closed-loop driving policy.

World models and future-aware driving. World models aim to provide anticipatory context by predicting future observations, latent states, or environment dynamics before action selection. Existing approaches span future-scene generation, video prediction, latent-state forecasting, spatio-temporal representation learning, and inverse-dynamics-guided future modeling (Hu et al., 2023a; Min et al., 2024; Hu et al., 2024; Zhang et al., 2026c;b;a). DriveWorld learns spatio-temporal representations from multi-camera driving data, while DrivingWorld focuses on controllable future video generation. More recent methods such as DeepSight and IDOL investigate compact latent-state prediction and inverse-dynamics-guided forecasting as alternatives to dense generative rollouts (Zhang et al., 2026c;b). These results highlight the value of anticipation for autonomous driving, but many world-model approaches remain centered on large-scale generative architectures, dense multi-step rollouts, or computationally intensive simulation pipelines. In some cases, generative driving world models scale to billions of parameters (Hu et al., 2023a).

Positioning of our work. Our work lies at the intersection of VLA driving, adaptive computation, and world-model-based anticipation. Unlike prior VLA approaches that apply reasoning uniformly across driving frames, we study adaptive reasoning in closed-loop autonomous driving. Unlike generative world-model methods that rely on dense future rollouts, we employ lightweight feature-level anticipation to provide short-term future context. The resulting framework integrates adaptive reasoning and future-aware representations within a unified VLA policy, treating both reasoning and anticipation as computational resources that can be deployed selectively according to driving complexity.

3 METHOD

3.1 OVERVIEW

We introduce ProActiveDrive, a VLA framework for closed-loop autonomous driving that combines adaptive reasoning with lightweight anticipation. As illustrated in Fig. 1, ProActiveDrive extends a SimLingo-style driving policy through a predictive fusion pipeline that enriches visual representations before they are processed by the language model. Specifically, image features are augmented with (i) spatial context from a LiDAR bird's-eye-view (BEV) representation and (ii) compact future features predicted by a DeltaTok-based forecasting module. In addition, an adaptive reasoning mechanism selectively invokes explicit reasoning when additional deliberation is expected to be beneficial. At time step t, the policy receives a front-view image $\boldsymbol{I}_t$, a navigation command $\boldsymbol{c}_t$, and ego-state information $\boldsymbol{s}_t$, and auxiliary sensor inputs when available. The driving policy then predicts future waypoints as:

$$\hat{\boldsymbol{y}}_t = \pi_\theta(\boldsymbol{I}_t, \boldsymbol{c}_t, \boldsymbol{s}_t). \tag{1}$$

Importantly, ProActiveDrive preserves the original output interface of the underlying VLA policy. The model continues to predict route and speed waypoints, which are subsequently tracked by low-

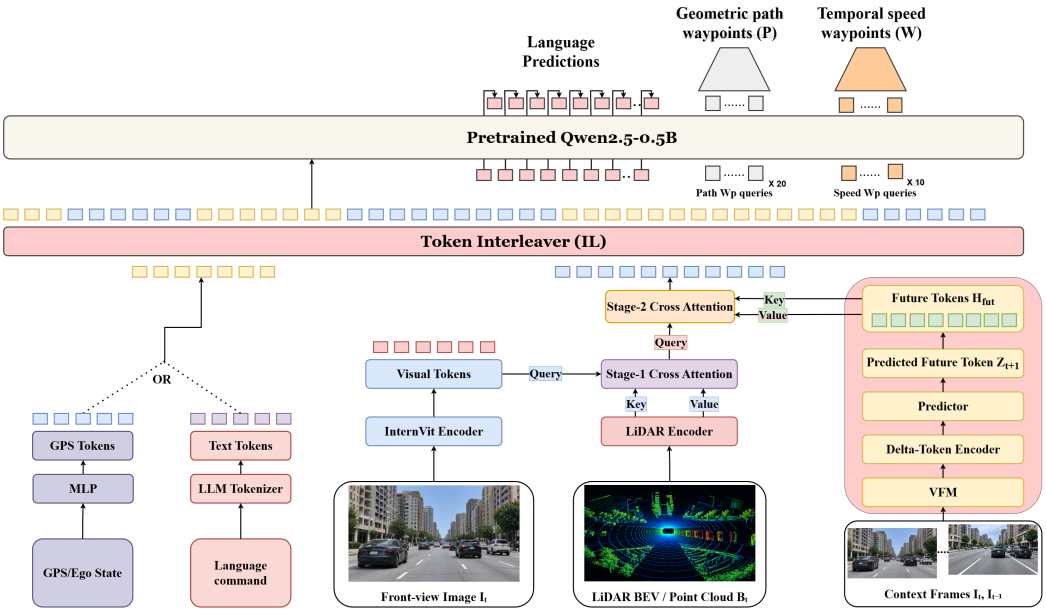

Figure 1: **Overview of ProActiveDrive.** A front-view image is encoded into visual tokens and enriched through a two-stage predictive fusion. Stage 1 integrates LiDAR features to provide the current scene spatial context, while Stage 2 incorporates the DeltaTok-predicted future representations for short-term anticipation. The fused tokens are interleaved with navigation and ego-state tokens and processed by a pretrained language model to jointly generate language outputs, geometric path waypoints, and speed waypoints.

level PID controllers. By augmenting the representation rather than the control head, ProActiveDrive enables future-aware and adaptive reasoning while remaining compatible with existing waypoint-based driving architectures.

3.2 Predictive-Fusion Encoder

The predictive-fusion encoder enriches image tokens with current scene structure and anticipated future dynamics before they enter the VLA policy. Let $\boldsymbol{H}_{\text{img}} \in \mathbb{R}^{N_v \times d}$ denote the N_v visual tokens produced by the InternViT encoder, with d the language-model hidden dimension. We encode the LiDAR-BEV occupancy map $\boldsymbol{B}_t$ with a BEV encoder f_{bev} and project the result to dimension d:

$$\boldsymbol{H}_{\text{bev}} = \text{Proj}_{\text{bev}}\big(f_{\text{bev}}(\boldsymbol{B}_t)\big). \tag{2}$$

Fusion then proceeds in two cross-attention stages. The visual tokens first attend to the BEV tokens, which injects spatial occupancy and scene layout. They then attend to the predicted future tokens $\boldsymbol{H}_{\text{fut}}$, which adds anticipated dynamics:

$$\boldsymbol{H}^{(1)} = \text{CrossAttn}\big(Q = \boldsymbol{H}_{\text{img}},\ K = V = \boldsymbol{H}_{\text{bev}}\big), \tag{3}$$

$$\boldsymbol{H}^{(2)} = \text{CrossAttn}\big(Q = \boldsymbol{H}^{(1)},\ K = V = \boldsymbol{H}_{\text{fut}}\big). \tag{4}$$

This ordering grounds the representation in the current scene geometry before enriching it with future context. The fused tokens $\boldsymbol{H}^{(2)} \in \mathbb{R}^{N_v \times d}$ replace the original image tokens in the VLA input sequence:

$$\boldsymbol{H}_{\text{VLA}} = \text{LLM}\big([\boldsymbol{H}^{(2)},\ \boldsymbol{H}_{\text{text}}]\big), \tag{5}$$

where $\boldsymbol{H}_{\text{text}}$ contains the driving command and prompt tokens. The SimLingo language model and control heads remain unchanged. The policy therefore reasons jointly over current observations, spatial structure, and short-term future context without explicit multi-step generative rollouts.

3.3 DELTATOK FUTURE FORECASTING

To provide anticipatory context without the cost of dense future generation, we adopt a frozen DeltaTok tokenizer (Kerssies et al., 2026), which represents visual change with compact latent tokens rather than RGB pixels. Let $\phi(\cdot)$ be a frozen foundation encoder that extracts feature maps from two frames $\boldsymbol{I}_i$ and $\boldsymbol{I}_j$. The tokenizer encodes the temporal change between them as

$$\boldsymbol{z}_{i\to j} = \mathrm{Enc}_\Delta\big(\phi(\boldsymbol{I}_i), \phi(\boldsymbol{I}_j)\big). \tag{6}$$

We write $\boldsymbol{z}_t = \boldsymbol{z}_{t-1\to t}$ for the delta token between consecutive frames. Future observations are unavailable during closed-loop inference, so a predictor P_ψ forecasts the next delta token from the K most recent ones:

$$\hat{\boldsymbol{z}}_{t+1} = P_\psi\big(\boldsymbol{z}_{t-K+1}, \ldots, \boldsymbol{z}_t\big). \tag{7}$$

The predicted token is decoded into future features, which are projected to the VLA hidden dimension d:

$$\hat{\boldsymbol{F}}_{t+1} = \mathrm{Dec}_\Delta\big(\phi(\boldsymbol{I}_t), \hat{\boldsymbol{z}}_{t+1}\big), \qquad \boldsymbol{H}_{\mathrm{fut}} = \mathrm{Proj}_{\mathrm{fut}}\big(\hat{\boldsymbol{F}}_{t+1}\big). \tag{8}$$

The resulting future tokens $\boldsymbol{H}_{\mathrm{fut}}$ compactly represent short-term scene evolution. The predictive-fusion encoder fuses them with the current observation.

DeltaTok remains frozen throughout driving-policy training. Only the predictor, BEV encoder, projection layers, and cross-attention fusion layers are optimized. During training the future frame is available, so the target token is computed directly as $\boldsymbol{z}^\star_{t+1} = \mathrm{Enc}_\Delta\big(\phi(\boldsymbol{I}_t), \phi(\boldsymbol{I}_{t+1})\big)$. The predictor is trained using a regression loss ($\rho(\cdot,\cdot)$) in the DeltaTok latent space:

$$\mathcal{L}_{\mathrm{pred}} = \rho\big(\hat{\boldsymbol{z}}_{t+1}, \boldsymbol{z}^\star_{t+1}\big). \tag{9}$$

3.4 ADAPTIVE THINKING

We cast reasoning allocation as a binary adaptive-computation problem. The model learns to distinguish routine frames (e.g., lane or car following), which can be handled through direct waypoint prediction, from decision-critical frames (e.g., intersections, pedestrians or cyclists, blocked lanes, overtaking, give-way situations, traffic lights, and stop signs) that benefit from explicit reasoning. Importantly, adaptive thinking modifies only the language supervision; the expert waypoint targets remain unchanged. We represent the supervision target as:

$$y_t = \begin{cases} \langle \mathrm{fast} \rangle\ \mathrm{Waypoints}, & \text{routine frame}, \\ \langle \mathrm{slow} \rangle\ r_t\ \mathrm{Waypoints}, & \text{critical frame}, \end{cases} \tag{10}$$

where r_t is a grounded reasoning trace generated before the waypoints. In fast mode, the model predicts waypoints directly. In slow mode, it first generates reasoning tokens and then predicts waypoints. Fast/slow labels are derived from driving-log heuristics based on hazard indicators, traffic-rule states, and interaction-critical situations. Grounded reasoning traces are automatically generated from scene attributes including nearby agents, traffic controls, distances, velocities, and ego motion.

3.5 TRAINING OBJECTIVE

The model is trained using a multi-task objective that combines language supervision, waypoint prediction, and future forecasting:

$$\mathcal{L} = \mathcal{L}_{\mathrm{lang}} + \mathcal{L}_{\mathrm{route}} + \mathcal{L}_{\mathrm{speed}} + \lambda_{\mathrm{pred}}\mathcal{L}_{\mathrm{pred}}, \tag{11}$$

where $\mathcal{L}_{\mathrm{route}}$ and $\mathcal{L}_{\mathrm{speed}}$ supervise route and speed waypoints, respectively, and $\mathcal{L}_{\mathrm{pred}}$ is the auxiliary DeltaTok forecasting loss. We set $\lambda_{\mathrm{pred}} = 0.2$ in all experiments. The language objective is the standard autoregressive next-token loss:

$$\mathcal{L}_{\mathrm{lang}} = -\sum_{i=1}^{|y_t|} \log p_\theta\left(y_{t,i} \mid y_{t,<i}, \boldsymbol{I}_t, c_t, \boldsymbol{s}_t\right). \tag{12}$$

The same objective supervises both fast-mode responses and slow-mode reasoning traces, allowing the model to learn when additional reasoning is beneficial while preserving a unified training formulation.

3.6 TWO-STAGE TRAINING

Starting from the SimLingo backbone and the pretrained DeltaTok tokenizer, we train the policy in two stages.

Stage 1: Predictive fusion. We first train the predictive-fusion module to align image, command, LiDAR-BEV, and predicted future representations for waypoint prediction. Trainable components include the DeltaTok predictor, BEV encoder, fusion and projection layers, and LoRA adapters on the language model.

Stage 2: Modality dropout and adaptive thinking. We then fine-tune the model with fusion dropout and adaptive-thinking supervision. For a subset of training samples, both BEV and future representations are jointly masked:

$$\tilde{\boldsymbol{H}}_{\mathrm{bev}} = m\boldsymbol{H}_{\mathrm{bev}}, \qquad \tilde{\boldsymbol{H}}_{\mathrm{fut}} = m\boldsymbol{H}_{\mathrm{fut}}, \qquad m \sim \mathrm{Bernoulli}(1-p), \tag{13}$$

with $p = 0.3$. This regularization discourages over-reliance on fused sensory features and preserves the utility of the command-conditioned pathway. During inference, the full fused representation is used. The same stage introduces grounded adaptive-thinking supervision, teaching the model when to respond directly and when explicit reasoning is beneficial, while keeping the control heads unchanged.

3.7 INFERENCE

At inference time, the BEV encoder extracts scene tokens from the current observation, while the DeltaTok predictor forecasts future tokens from past frames. These representations are fused with image, command, and ego-state embeddings through two-stage cross-attention and processed by the language model. Depending on the situation, the model generates either a direct response or an explicit reasoning trace, and the predicted waypoints are executed by a PID controller.

4 EXPERIMENTS

4.1 DATASETS AND METRICS

Datasets. We evaluate ProActiveDrive on Bench2Drive-220 (Jia et al., 2024), which contains 220 standardized closed-loop routes covering diverse interactive driving scenarios. To further evaluate robustness under distribution shift, we conduct experiments on Fail2Drive (Gerstenecker et al., 2026), which provides paired in-distribution and generalization scenarios with unseen spatio-temporal configurations.

Metrics. For Bench2Drive, we report Driving Score (DS), Success Rate (SR), and the official multi-ability success rates for Merging, Overtaking, Emergency Braking, Give Way, and Traffic Sign handling. For Fail2Drive, we report DS, SR, and their Harmonic Mean (HM). We additionally report inference throughput for the efficiency analysis.

Table 1: Closed-loop comparison with representative methods on Bench2Drive-220. Results for ProActiveDrive are reported as mean ± standard deviation. Best in **bold**, second-best underlined.

Method	Cam.	LiDAR	SR (%) ↑	DS ↑
UniAD-Base Hu et al. (2023b)	6×	×	16.36	45.81
DriveTransformer Jia et al. (2025)	6×	×	35.01	63.46
ReCogDrive Li et al. (2026b)	6×	×	45.45	71.36
Hydra-NeXt Li et al. (2025)	2×	×	50.00	73.86
DiffusionDrive Liao et al. (2025)	3×	✓	52.72	77.68
ORION Fu et al. (2025)	6×	×	54.62	77.74
GeRo Yasarla et al. (2026)	6×	×	60.10	81.90
TF++ Jaeger et al. (2023)	1×	✓	67.27	84.21
SimLingo Renz et al. (2025)	1×	×	67.27	86.07
R2SE Liu et al. (2025)	1×	✓	<u>69.54</u>	86.28
HiP-AD Tang et al. (2025)	6×	×	69.09	86.77
DeLL Du et al. (2026)	1×	✓	68.63	<u>86.86</u>
ProActiveDrive	1×	✓	**71.30 ± 2.67**	**88.20 ± 1.04**

4.2 MAIN RESULTS

Closed-loop planning performance. Table 1 compares ProActiveDrive with different baselines on Bench2Drive-220 for which the corresponding multi-ability evaluation is also available. Our method achieves the best overall performance, reaching a Driving Score of 88.20 ± 1.04 and a Success Rate of $71.30 \pm 2.67\%$. Compared with the closest architectural baseline, SimLingo, ProActiveDrive improves DS by 2.13 points and SR by 4.03 percentage points. These results indicate that augmenting a compact single-camera VLA policy with LiDAR-based spatial context and predictive future representations yields more reliable long-horizon decision making.

Multi-ability performance. Table 2 further breaks down performance across the five official Bench2Drive abilities. ProActiveDrive obtains the second-highest mean score (68.70%), narrowly trailing DeLL (68.90%), while matching or exceeding the strongest baselines in several interaction-centered behaviors. In particular, ProActiveDrive improves over SimLingo by $+5.32$ points on Merging, $+9.29$ points on Overtaking, and $+10.00$ points on GiveWay, resulting in a $+2.69$ point gain in overall mean ability score. The largest improvements arise in scenarios requiring anticipation of other agents and future traffic evolution, suggesting that the predicted future representations primarily benefit interactive planning. Emergency Braking and Traffic Sign handling remain challenging, indicating room for further improvements in safety-critical and rule-based behaviors.

Table 2: Multi-ability success rate (%) on Bench2Drive-220 for the same comparison methods reported in Table 1. Best in **bold**, second-best underlined.

Method	Cam.	LiDAR	Merge ↑	Overtake ↑	EmBrake ↑	GiveWay ↑	TSign ↑	Mean ↑
UniAD-Base Hu et al. (2023b)	6×	×	14.10	17.78	21.67	10.00	14.21	15.55
DriveTransformer Jia et al. (2025)	6×	×	17.57	35.00	48.36	40.00	52.10	38.60
ReCogDrive Li et al. (2026b)	6×	×	29.73	20.00	69.09	20.00	71.34	42.03
Hydra-NeXt Li et al. (2025)	2×	×	40.00	64.44	61.67	50.00	50.00	53.22
DiffusionDrive Liao et al. (2025)	3×	✓	50.63	26.67	68.33	50.00	76.32	54.38
ORION Fu et al. (2025)	6×	×	25.00	71.11	78.33	30.00	69.15	54.72
GeRo Yasarla et al. (2026)	6×	×	40.06	78.24	87.32	50.00	76.83	66.49
TF++ Jaeger et al. (2023)	1×	✓	58.75	57.77	83.33	40.00	82.11	64.39
SimLingo Renz et al. (2025)	1×	×	53.78	67.41	81.67	50.00	77.20	66.01
R2SE Liu et al. (2025)	1×	✓	53.33	61.25	**90.00**	50.00	**84.21**	67.76
HiP-AD Tang et al. (2025)	6×	×	50.00	**84.44**	83.33	40.00	72.10	65.98
DeLL Du et al. (2026)	1×	✓	**61.25**	62.22	80.00	**60.00**	81.05	**68.90**
ProActiveDrive	1×	✓	59.10	76.70	80.00	**60.00**	67.70	68.70

Generalization under distribution shift. Table 3 evaluates transfer to the Fail2Drive benchmark, which introduces driving configurations not encountered during training. ProActiveDrive achieves the highest harmonic mean in both the in-distribution (81.40) and generalization (63.90) settings, outperforming the closest baseline, SimLingo, by $+0.50$ and $+1.70$ points, respectively. The larger improvement under distribution shift suggests that future-aware representations provide a more robust planning signal when encountering previously unseen scene configurations.

Table 3: Closed-loop generalization on Fail2Drive. HM denotes the harmonic mean of DS and SR.

	In-Distribution			Generalization		
Method	**DS ↑**	**SR ↑**	**HM ↑**	**DS ↑**	**SR ↑**	**HM ↑**
TCP Wu et al. (2022)	24.7	39.1	30.3	24.5	31.4	27.5
UniAD Hu et al. (2023b)	47.5	36.3	41.2	44.0	27.6	33.9
ORION Fu et al. (2025)	53.0	52.0	52.5	51.2	46.0	48.5
HiP-AD Tang et al. (2025)	74.1	70.7	72.4	67.1	**56.7**	61.5
SimLingo Renz et al. (2025)	82.6	**79.3**	80.9	71.7	55.0	62.2
ProActiveDrive	**83.95**	79.00	**81.40**	**73.96**	56.25	**63.90**

Inference efficiency. In addition to driving performance, we evaluate the computational effect of adaptive reasoning. Table 4 compares direct SimLingo inference, ProActiveDrive with always-on

CoT, and ProActiveDrive using adaptive thinking. Adaptive thinking reaches approximately $2.96\times$ the effective throughput of always-on CoT. This supports the motivation for selective reasoning: explicit language reasoning can be retained for difficult situations without imposing its autoregressive cost at every timestep.

Table 4: Inference efficiency of different reasoning strategies.

Setting	Mean Ratio ↑	Relative Speed
SimLingo, no CoT	$0.0244\times$	$1.63\times$
Full CoT	$0.0150\times$	$1.00\times$
Adaptive thinking	$\mathbf{0.0443}\times$	$\mathbf{2.96}\times$

4.3 Ablation Study

We analyze the contributions of the two central components of ProActiveDrive: grounded adaptive reasoning and future-aware prediction.

Role of adaptive reasoning. Table 5 compares grounded adaptive reasoning against conventional template-based CoT supervision. Grounded adaptive reasoning improves the mean multi-ability score from 65.6 to 68.7 (+3.1 points). The largest gains occur in Overtaking (+12.3) and GiveWay (+10.0), while effects on other abilities are modest. This pattern suggests that adaptive reasoning is most beneficial in interaction-centric scenarios that require explicit interpretation of surrounding agents, intentions, and right-of-way rules, rather than uniformly affecting all driving behaviors.

Table 5: Ablation of grounded adaptive-thinking supervision.

Setting	Over.	Merge	EmBr.	GiveW.	TSign	Mean
Full CoT	64.4	61.2	81.7	50.0	70.5	65.6
Adaptive thinking	**76.7**	59.1	80.0	**60.0**	67.7	**68.7**
Gain	**+12.3**	-2.1	-1.7	**+10.0**	-2.8	**+3.1**

Role of grounded supervision. Table 6 evaluates the effect of the grounding and refinement process used to construct adaptive reasoning labels. Grounding increases correct yield intent from 33% to 88% while reducing over-yielding from 79% to 20%. The resulting supervision exhibits strong object-grounding accuracy (90%) and red-light-to-stop consistency (94%), yielding a total of 170,241 grounded reasoning annotations. These improvements indicate that the supervision better aligns reasoning traces with scene dynamics and traffic rules, providing a plausible explanation for the gains observed in interaction-heavy driving abilities.

Table 6: Quality improvement of the grounded CoT supervision.

Metric	Before	After	Change
Correct yield intent	33%	**88%**	**+55 pp**
Over-yield rate	79%	**20%**	-59 **pp**
Object grounding	–	**90%**	–
Red-light → stop consistency	–	**94%**	–

How much future horizon is needed? Table 7 evaluates whether extending future prediction beyond the immediate next step improves closed-loop driving. Surprisingly, predicting only the next latent state, $\boldsymbol{z}_{t+1}$, achieves the highest driving score (88.20 ± 1.04), outperforming a four-horizon variant by 2.34 points. Although multi-horizon prediction yields slightly higher throughput, it does not translate into better driving performance. We hypothesize that the immediate future provides the most actionable planning signal, whereas more distant predictions become increasingly uncertain and may dilute information relevant to the current decision.

Table 7: Ablation of the future-prediction horizon.

Setting	Future Representation	DS ↑	Mean Ratio ↑
Single-step	$\boldsymbol{z}_{t+1}$	**88.20 ± 1.04**	0.0443×
Multi-horizon	$\{\boldsymbol{z}_{t+1}, \ldots, \boldsymbol{z}_{t+4}\}$	85.86 ± 0.52	**0.0477**×

5 Conclusion

We introduced ProActiveDrive, a framework that combines grounded adaptive reasoning with future-aware representations for VLA autonomous driving. Our results show that improving the quality and deployment of reasoning can yield substantial gains in closed-loop performance, even without increasing reasoning frequency or complexity. In particular, grounded reasoning supervision improves behavior in interaction-heavy scenarios, while selective reasoning reduces inference cost by invoking deliberation only when needed. We further find that compact prediction of the immediate future provides a stronger planning signal than extending prediction farther into the horizon. Across multiple benchmarks, these components translate into improved driving performance, robustness under distribution shift, and computational efficiency. More broadly, our findings suggest that effective language-action alignment depends not only on model architecture but also on how reasoning is supervised, grounded, and integrated into the control loop. Closing the remaining generalization gap in long-tail driving scenarios remains an important direction for future work.

References

Sijia Chen and Baochun Li. Toward adaptive reasoning in large language models with thought rollback. In *ICML*, pp. 7033–7056, 2024.

Xiaoxue Cheng, Junyi Li, Zhenduo Zhang, Xinyu Tang, Wayne Xin Zhao, Xinyu Kong, and Zhiqiang Zhang. Incentivizing dual process thinking for efficient large language model reasoning. In *NeurIPS*, 2025.

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

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.

Jiayuan Du, Yuebing Song, Yiming Zhao, Xianghui Pan, Jiawei Lian, Yuchu Lu, Liuyi Wang, Chengju Liu, and Qijun Chen. Deconfounded lifelong learning for autonomous driving via dynamic knowledge spaces. In *ECCV*, 2026.

Haoyu Fu, Diankun Zhang, Zongchuang Zhao, Jianfeng Cui, Dingkang Liang, Chong Zhang, Dingyuan Zhang, Hongwei Xie, Bing Wang, and Xiang Bai. ORION: A holistic end-to-end autonomous driving framework by vision-language instructed action generation. In *ICCV*, 2025.

Simon Gerstenecker, Andreas Geiger, and Katrin Renz. Fail2drive: Benchmarking closed-loop driving generalization, 2026.

Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. GAIA-1: A generative world model for autonomous driving, 2023a.

Xiaotao Hu, Mingkai Jia, Xiaoyang Guo, Qian Zhang, Xiao-xiao Long, and Wei Yin. DrivingWorld: Constructing world model for autonomous driving via video GPT, 2024.

Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, Lewei Lu, Xiaosong Jia, Qiang Liu, Jifeng Dai, Yu Qiao, and Hongyang Li. Planning-oriented autonomous driving. In *CVPR*, pp. 17853–17862, 2023b.

Bernhard Jaeger, Kashyap Chitta, and Andreas Geiger. Hidden biases of end-to-end driving models. In *ICCV*, pp. 8240–8249, 2023.

Metod Jazbec, Alexander Timans, Tin Hadži Veljković, Kaspar Sakmann, Dan Zhang, Christian A. Naesseth, and Eric Nalisnick. Fast yet safe: Early-exiting with risk control. In *NeurIPS*, 2024.

Feiyang Jia, Lin Liu, Ziying Song, Caiyan Jia, Hangjun Ye, Xiaoshuai Hao, and Long Chen. DriveWorld-VLA: Unified latent-space world modeling with vision-language-action for autonomous driving, 2026.

Xiaosong Jia, Zhenjie Yang, Qifeng Li, Zhiyuan Zhang, and Junchi Yan. Bench2drive: Towards multi-ability benchmarking of closed-loop end-to-end autonomous driving. In *NeurIPS*, 2024.

Xiaosong Jia, Junqi You, Zhiyuan Zhang, and Junchi Yan. Drivetransformer: Unified transformer for scalable end-to-end autonomous driving. In *ICLR*, 2025.

Tommie Kerssies, Gabriele Berton, Ju He, Qihang Yu, Wufei Ma, Daan de Geus, Gijs Dubbelman, and Liang-Chieh Chen. A frame is worth one token: Efficient generative world modeling with delta tokens. In *CVPR*, 2026.

Fanjie Kong, Yitong Li, Weihuang Chen, Chen Min, Yizhe Li, Zhiqiang Gao, Haoyang Li, Zhongyu Guo, and Hongbin Sun. VLR-Driver: Large vision-language-reasoning models for embodied autonomous driving. In *ICCV*, pp. 26966–26976, 2025.

Pingzhi Li, Bairu Hou, Yun Zhu, Yihao Feng, Ke Ye, Tao Lei, Zhifeng Chen, Tianlong Chen, and Xianzhi Du. Adaptive thinking: Large language models know when to think in latent space. In *International Conference on Learning Representations*, 2026a.

Yongkang Li, Kaixin Xiong, Xiangyu Guo, Fang Li, Sixu Yan, Gangwei Xu, Lijun Zhou, Long Chen, Haiyang Sun, Bing Wang, Kun Ma, Guang Chen, Hangjun Ye, Wenyu Liu, and Xinggang Wang. Recogdrive: A reinforced cognitive framework for end-to-end autonomous driving. In *ICLR*, 2026b.

Zhenxin Li, Shihao Wang, Shiyi Lan, Zhiding Yu, Zuxuan Wu, and Jose M. Alvarez. Hydra-next: Robust closed-loop driving with open-loop training. In *ICCV*, pp. 27305–27314, 2025.

Bencheng Liao, Shaoyu Chen, Haoran Yin, Bo Jiang, Cheng Wang, Sixu Yan, Xinbang Zhang, Xiangyu Li, Ying Zhang, Qian Zhang, and Xinggang Wang. Diffusiondrive: Truncated diffusion model for end-to-end autonomous driving. In *CVPR*, pp. 12037–12047, 2025.

George Ling, Lijin Yang, Hao Yang, and Zhongzhan Huang. BLUE: Toward better language use in efficient vision-language-action models for autonomous driving. In *EMNLP*, 2026. Accepted to EMNLP 2026 Main Conference.

Haochen Liu, Tianyu Li, Haohan Yang, Li Chen, Caojun Wang, Ke Guo, Haochen Tian, Hongchen Li, Hongyang Li, and Chen Lv. Reinforced refinement with self-aware expansion for end-to-end autonomous driving, 2025.

Xiang Liu, Xuming Hu, Xiaowen Chu, and Eunsol Choi. DiffAdapt: Difficulty-adaptive reasoning for token-efficient llm inference. In *ICLR*, 2026.

Chen Min, Dawei Zhao, Liang Xiao, Jian Zhao, Xinli Xu, Zheng Zhu, Lei Jin, Jianshu Li, Yulan Guo, Junliang Xing, Liping Jing, Yiming Nie, and Bin Dai. Driveworld: 4d pre-trained scene understanding via world models for autonomous driving. In *CVPR*, pp. 15522–15533, 2024.

Katrin Renz, Long Chen, Elahe Arani, and Oleg Sinavski. Simlingo: Vision-only closed-loop autonomous driving with language-action alignment. In *CVPR*, pp. 11993–12003, 2025.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *NeurIPS*, 2022.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *ICLR*, 2025.

Yingqi Tang, Zhuoran Xu, Zhaotie Meng, and Erkang Cheng. HiP-AD: Hierarchical and multi-granularity planning with deformable attention for autonomous driving in a single decoder. In *ICCV*, pp. 25605–25615, 2025.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *ICLR*, 2023.

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

Penghao Wu, Xiaosong Jia, Li Chen, Junchi Yan, Hongyang Li, and Yu Qiao. Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. In *NeurIPS*, 2022.

Zhenya Yang, Zhe Liu, Yuxiang Lu, Liping Hou, Chenxuan Miao, Siyi Peng, Bailan Feng, Xiang Bai, and Hengshuang Zhao. Geniedrive: Towards physics-aware driving world model with 4d occupancy guided video generation. In *CVPR*, pp. 35680–35690, 2026.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *NeurIPS*, 2023.

Rajeev Yasarla, Deepti Hegde, Shizhong Han, Hsin-Pai Cheng, Yunxiao Shi, Meysam Sadeghigooghari, Shweta Mahajan, Apratim Bhattacharyya, Litian Liu, Risheek Garrepalli, Thomas Svantesson, Fatih Porikli, and Hong Cai. Generative scenario rollouts for end-to-end autonomous driving, 2026.

Bozhou Zhang, Nan Song, Yuang Wang, Jiankang Deng, Xiatian Zhu, and Li Zhang. See tomorrow, act today: Foresight-driven autonomous driving, 2026a.

Chenghao Zhang, Timin Li, and Dongmei Li. IDOL: Inverse-dynamics-guided future prediction for end-to-end autonomous driving, 2026b.

Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. Adaptthink: Reasoning models can learn when to think. In *EMNLP*, 2025.

Lingjun Zhang, Changjie Wu, Linzhe Shi, Jiangyang Li, Jiaxin Liu, Lei Yang, Hang Zhang, Mu Xu, and Hong Wang. Deepsight: Long-horizon world modeling via latent states prediction for end-to-end autonomous driving. In *ICML*, 2026c.

Xingcheng Zhou, Xuyuan Han, Feng Yang, Yunpu Ma, Volker Tresp, and Alois Knoll. OpenDriveVLA: Towards end-to-end autonomous driving with large vision language action model. In *AAAI*, volume 40, pp. 13782–13790, 2026.

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

Julian Zimmerlin, Jens Beisswenger, Bernhard Jaeger, Andreas Geiger, and Kashyap Chitta. Hidden biases of end-to-end driving datasets, 2024.

ACKNOWLEDGMENTS

Omitted for double-blind review.

Appendix: Supplementary Material

A Training and Evaluation Details

This section provides additional implementation details for training ProActiveDrive, including the optimization strategy, distributed training configuration, predictive-fusion fine-tuning, DeltaTok training, and closed-loop evaluation setup.

A.1 Optimization Strategy

The driving policy is initialized from the pretrained SimLingo InternVL2-1B checkpoint. The backbone consists of an InternViT visual encoder and a Qwen2-0.5B language model. We use LoRA adaptation for the language model with rank $r = 32$, scaling factor $\alpha = 64$, and LoRA dropout of 0.1.

We optimize the model using AdamW with

$$\beta_1 = 0.9, \qquad \beta_2 = 0.999, \qquad \lambda_{\text{wd}} = 0.1. \tag{14}$$

We use two learning-rate groups to account for the different initialization of pretrained and newly introduced modules. Parameters inherited from the pretrained VLA are optimized with

$$\eta_{\text{base}} = 5 \times 10^{-6}, \tag{15}$$

whereas newly initialized modules use a ten-times larger learning rate,

$$\eta_{\text{new}} = 5 \times 10^{-5}. \tag{16}$$

The second group includes the BEV encoder, DeltaTok Predictor, two-stage cross-attention modules, target-point encoder, and driving prediction heads. The frozen DeltaTok tokenizer and its frozen DINOv3 backbone are excluded from the optimizer.

To avoid unnecessarily regularizing normalization parameters, biases, and learned scalar gates, one-dimensional parameters are excluded from weight decay.

The learning rate follows a OneCycle schedule:

$$\eta_t = \text{OneCycleLR}\left(\eta_{\max}, t; p_{\text{warm}} = 0.05\right), \tag{17}$$

where the first 5% of the optimization schedule is used for learning rate warm-up. The scheduler is updated after every optimizer step.

We train using BF16 mixed precision. Gradients are clipped using the global L_2 norm with

$$\|\nabla_\theta \mathcal{L}\|_2 \leq 0.3. \tag{18}$$

A.2 Distributed Training Configuration

The main VLA model is trained using four NVIDIA A100 GPUs with $80\,\text{GB}$ of memory per GPU. Training uses Distributed Data Parallel (DDP), with one model replica on each GPU.

The batch size is 5 samples per GPU. Therefore, with four distributed workers, the effective global batch size is

$$B_{\text{global}} = B_{\text{GPU}} \times N_{\text{GPU}} = 5 \times 4 = 20. \tag{19}$$

No additional gradient accumulation is used for the main VLA training. Each distributed optimization step therefore processes 20 samples globally.

We use eight data-loading workers per training process. Synchronized BatchNorm is disabled because the LiDAR-BEV branch is conditionally executed depending on the available modality and training sample. The DDP configuration enables unused-parameter detection to support these conditional computational paths.

All experiments use random seed 42.

Table 8 summarizes the main training configuration.

Table 8: Training configuration of the predictive-fusion VLA policy.

Configuration	Value
Training hardware	$4\times$ NVIDIA A100 80 GB
Distributed strategy	DDP
Precision	BF16 mixed
Batch size / GPU	5
Global batch size	20
Data workers / process	8
Optimizer	AdamW
Base learning rate	5×10^{-6}
New-module learning rate	5×10^{-5}
Weight decay	0.1
AdamW betas	(0.9, 0.999)
LR scheduler	OneCycleLR
Warm-up fraction	5%
Gradient clipping	0.3
LoRA rank	32
LoRA α	64
LoRA dropout	0.1
Random seed	42

A.3 Predictive-Fusion Training

The predictive-fusion policy is trained progressively rather than introducing all regularization mechanisms from the beginning. We first initialize the model from the original SimLingo checkpoint and train the new predictive-fusion architecture without modality dropout. This stage allows the newly initialized BEV encoder, DeltaTok Predictor, projection layers, and cross-attention modules to learn how to interact with the pretrained visual-language representation.

The first fusion stage uses InternViT visual tokens as queries and LiDAR-BEV tokens as keys and values. The second stage uses the LiDAR-enriched visual representation as queries and the predicted DeltaTok future representation as keys and values. Both cross-attention stages contain two blocks. The DeltaTok Predictor contains six transformer layers with twelve attention heads.

The DeltaTok tokenizer itself is not optimized during VLA training. Instead, it remains frozen and provides the latent representation used by the trainable Predictor. The Predictor receives four past context frames ending at the current observation. Consecutive context frames are separated by approximately 0.25 s.

In addition to the language and waypoint objectives, we optimize an auxiliary future-prediction loss:

$$\mathcal{L} = \mathcal{L}_{\text{lang}} + \mathcal{L}_{\text{route}} + \mathcal{L}_{\text{speed}} + 0.2\,\mathcal{L}_{\text{pred}}. \tag{20}$$

The first three objectives have unit weight. The future-prediction term supervises the Predictor against the teacher-forced future DeltaTok target available during training. Importantly, the future frame used to construct this target is available only during training and is never required during closed-loop inference.

The initial predictive-fusion training is performed without modality dropout. After convergence, we use the checkpoint at approximately 125K optimization steps as initialization for the subsequent modality-dropout stage.

A.4 MODALITY-DROPOUT FINE-TUNING

After predictive-fusion training, we perform a second fine-tuning stage designed to prevent the driving policy from becoming overly dependent on the richer BEV and future-prediction signals.

We warm-start from the converged predictive-fusion checkpoint and train for an additional five epochs with joint fusion dropout. For every training sample, a Bernoulli variable determines whether the auxiliary fusion information is retained:

$$m_t \sim \mathrm{Bernoulli}(1 - p_{\mathrm{drop}}), \qquad p_{\mathrm{drop}} = 0.3. \tag{21}$$

The BEV and future branches are then jointly masked according to

$$\tilde{\boldsymbol{H}}_{\mathrm{BEV}} = m_t \boldsymbol{H}_{\mathrm{BEV}}, \qquad \tilde{\boldsymbol{H}}_{\mathrm{fut}} = m_t \boldsymbol{H}_{\mathrm{fut}}. \tag{22}$$

Thus, approximately 30% of the corresponding training samples are optimized without access to the auxiliary fusion representation. This encourages the policy to preserve useful information in the original camera, navigation-command, and target-point pathways rather than learning a shortcut through the additional sensor branches.

Modality dropout is used only during training. At evaluation time,

$$m_t = 1, \tag{23}$$

and the complete LiDAR and future-prediction representations are provided to the model.

A.5 DELTATOK FINETUNING

DeltaTok is finetuned independently from the driving policy and is frozen before being integrated into the VLA. We use a frozen DINOv3 ViT-B/16 backbone and optimize the DeltaTok encoder and decoder in feature space.

The standalone DeltaTok models use AdamW with a learning rate of

$$\eta_{\Delta} = 1 \times 10^{-4}, \tag{24}$$

and a 1000-step warm-up. Training is performed using BF16 precision for $20{,}000$ optimizer steps.

For the single-horizon model, the physical batch size is 16, with gradient accumulation over two iterations. For the multi-horizon model, the increased memory requirement of simultaneously processing four future targets reduces the physical batch size to 8, while gradient accumulation is increased to four iterations.

The multi-horizon configuration therefore uses

$$B_{\mathrm{eff}} = 8 \times 4 = 32 \tag{25}$$

samples per optimization update in the DeltaTok training procedure.

The multi-horizon variant additionally uses weight decay 10^{-2} and gradient clipping at 10^{-2}. It is warm-started from the single-horizon checkpoint, while the horizon-specific embeddings are newly initialized.

A.6 CLOSED-LOOP EVALUATION INFRASTRUCTURE

Training and closed-loop evaluation are performed on separate GPU configurations. The model is trained using four NVIDIA A100 GPUs with $80\,\mathrm{GB}$ memory per GPU, while the closed-loop CARLA evaluation is executed using four NVIDIA A100 GPUs with $40\,\mathrm{GB}$ memory per GPU.

The evaluation GPUs are used to parallelize closed-loop route execution. Each evaluation process runs the trained policy together with the CARLA evaluation stack, while the resulting route statistics are aggregated to compute Driving Score (DS), Success Rate (SR), and the corresponding multi-ability metrics.

No training-time modality dropout is active during evaluation. The DeltaTok tokenizer remains frozen, and only past and current observations are available to the future Predictor. In particular, the teacher-forced future frame used by the auxiliary prediction objective during training is not available to the model during closed-loop evaluation.

Table 9: Standalone DeltaTok finetuning configuration.

Configuration	**Single**	**Multi-Horizon**
Future horizons	1	4
Optimizer	AdamW	AdamW
Learning rate	10^{-4}	10^{-4}
Warm-up	1,000	1,000
Training steps	20,000	20,000
Batch size	16	8
Grad. accumulation	2	4
Multi-horizon effective batch	–	32
Precision	BF16	BF16
DINOv3 backbone	Frozen	Frozen

Table 10: Compute infrastructure used for training and evaluation.

Stage	**GPUs**	**Memory / GPU**
Training	4× NVIDIA A100	80 GB
Closed-loop evaluation	4× NVIDIA A100	40 GB

B Additional DeltaTok Finetuning Details

The DeltaTok future predictor is fine-tuned on front-view RGB clips from Bench2Drive. The training split contains 250 clips comprising 77,359 frames, while the validation split contains 15 clips and 4,041 frames, with no clip overlap between the two splits.

Each image is center-cropped and resized to 512×512 before being processed by a frozen DINOv3 ViT-B/16 visual encoder. The encoder produces a 32×32 grid of $N = 1024$ patch tokens, each with feature dimension $D = 768$. DeltaTok is trained directly in this frozen feature space, allowing the predictor to model short-term scene evolution without requiring pixel-space generation.

For the single-horizon configuration used in the final driving model, the predictor learns to estimate the representation of the immediate future observation from the current visual features. The visual backbone remains frozen throughout finetuning, while only the future-prediction components are optimized. The resulting checkpoint reaches approximately 0.00238 validation reconstruction loss.

This single-horizon configuration is used in the final ProActiveDrive model because it provides a compact representation of imminent scene evolution while avoiding the additional complexity associated with simultaneously exposing the driving policy to multiple future horizons.

C Multi-Future-Frame Prediction

The main model uses a single near-future DeltaTok representation to provide anticipatory information to the driving policy. In this supplementary experiment, we investigate whether exposing the VLA to multiple future horizons provides additional benefits. Specifically, we extend the single-horizon DeltaTok formulation to predict representations associated with four future frames spanning approximately one second.

This experiment addresses two separate questions. First, can DeltaTok represent scene evolution across multiple temporal horizons? Second, does providing these additional future representations improve closed-loop driving compared with using only the immediate future?

C.1 MULTI-HORIZON DELTATOK

The single-horizon DeltaTok model represents the visual transition between the current frame and a near-future frame using a compact delta token. The multi-horizon extension increases the temporal window to five frames, corresponding to the current observation and four future horizons:

$$\{\boldsymbol{I}_t, \boldsymbol{I}_{t+1}, \boldsymbol{I}_{t+2}, \boldsymbol{I}_{t+3}, \boldsymbol{I}_{t+4}\}. \tag{26}$$

The temporal spacing between consecutive horizons is approximately $\Delta t = 0.25$ s. Therefore, the four future targets correspond to approximately

$$\mathcal{T}_{\text{fut}} = \{0.25,\ 0.5,\ 0.75,\ 1.0\}\ \text{s}. \tag{27}$$

All frames are encoded by the same frozen DINOv3 ViT-B/16 visual foundation model:

$$\boldsymbol{F}_{t+k} = \phi(\boldsymbol{I}_{t+k}), \qquad k \in \{0, \ldots, 4\}, \tag{28}$$

where $\boldsymbol{F}_{t+k} \in \mathbb{R}^{N \times 768}$ denotes the corresponding DINOv3 patch representation.

The multi-horizon DeltaTok model uses a shared delta representation together with horizon-specific embeddings. Let $\boldsymbol{z}_\Delta$ denote the compact delta token describing the visual transition over the temporal window. For each horizon k, a learned temporal embedding $\boldsymbol{e}_k$ conditions the decoder:

$$\hat{\boldsymbol{F}}_{t+k} = \text{Dec}_\Delta\left(\boldsymbol{F}_t,\ \boldsymbol{z}_\Delta + \boldsymbol{e}_k\right), \qquad k = 1, \ldots, 4. \tag{29}$$

Consequently, a single compact temporal representation can be decoded into multiple horizon-specific future feature maps:

$$\boldsymbol{z}_\Delta \longrightarrow \left\{\hat{\boldsymbol{F}}_{t+1}, \hat{\boldsymbol{F}}_{t+2}, \hat{\boldsymbol{F}}_{t+3}, \hat{\boldsymbol{F}}_{t+4}\right\}. \tag{30}$$

The horizon embeddings allow the shared decoder to distinguish the temporal offset associated with each prediction rather than treating all future targets identically.

C.2 TRAINING CONFIGURATION

The multi-horizon model is initialized from the single-horizon DeltaTok checkpoint. The Delta encoder and decoder are therefore warm-started from the model already adapted to the Bench2Drive visual domain, while the horizon-specific embeddings are initialized separately.

The DINOv3 ViT-B/16 backbone remains frozen throughout training. Only the DeltaTok encoder, decoder, and newly introduced horizon embeddings are optimized. We use the same feature-space reconstruction objective as for the single-horizon model.

For $K = 4$ future horizons, the objective is

$$\mathcal{L}_{\text{multi}} = \frac{1}{K} \sum_{k=1}^{K} \left\| \hat{\boldsymbol{F}}_{t+k} - \boldsymbol{F}_{t+k} \right\|_2^2, \qquad K = 4. \tag{31}$$

Thus, supervision is performed entirely in the frozen DINOv3 feature space rather than RGB space.

The main training configuration is summarized in Table 11.

C.3 FUTURE-FEATURE RECONSTRUCTION

We first evaluate whether the multi-horizon representation captures meaningful temporal changes independently of the driving policy. We compare the predicted DINOv3 features against the ground-truth features of the corresponding future frames using cosine similarity.

Because consecutive driving frames are visually similar, simply copying the current-frame representation already provides a relatively strong baseline. We therefore additionally compare against a static predictor defined as

$$\hat{\boldsymbol{F}}_{t+k}^{\text{static}} = \boldsymbol{F}_t. \tag{32}$$

Table 11: Training configuration of the multi-horizon DeltaTok model.

Configuration	**Value**
Visual backbone	DINOv3 ViT-B/16
Backbone status	Frozen
Input resolution	512×512
Feature dimension	768
Number of horizons	4
Frames per window	5
Temporal spacing	≈ 0.29 s
Maximum horizon	≈ 1.14 s
Training steps	20,000
Learning rate	1×10^{-4}
Warmup	1,000 steps
Batch size	8
Gradient accumulation	4
Effective batch size	32

Table 12: Cosine similarity between predicted and ground-truth future DINOv3 features. The static baseline directly reuses the current-frame features.

Model	$+0.25$**s**	$+0.5$**s**	$+0.75$**s**	$+1.00$**s**
Single-horizon	**0.990**	–	–	–
Multi-horizon	0.974	**0.966**	**0.970**	**0.983**
Static baseline	0.957	0.932	0.930	0.923
Multi − Static	+0.017	+0.034	+0.040	+0.060

This baseline is important because a high absolute similarity alone does not necessarily demonstrate successful temporal prediction. The relevant quantity is whether the predicted representation captures future change better than simply reusing the present scene.

Table 12 reports the results over the held-out Bench2Drive clips.

The multi-horizon model remains above the static baseline at every temporal offset. More importantly, the margin over the static baseline increases with prediction horizon, from $+0.017$ at 0.29 s to $+0.060$ at 1.14 s. This indicates that the learned representation captures temporal changes beyond the high visual similarity naturally present between neighboring driving frames.

At the shortest horizon, the specialized single-horizon model achieves a cosine similarity of 0.990, compared with 0.974 for the multi-horizon model. This result is expected because the single-horizon variant dedicates its complete predictive capacity to one immediate future target, whereas the multi-horizon model shares its representation across four temporal offsets.

C.4 Qualitative Multi-Frame Prediction

We additionally inspect the predicted future representations qualitatively. Since DeltaTok operates in feature space, the predicted DINOv3 features are mapped back to RGB only for visualization using a frozen reconstruction head. These visualizations should therefore be interpreted primarily in terms of scene geometry and agent placement rather than fine image texture.



Figure 2: Qualitative DeltaTok finetuning example 1.

For each example, we visualize the current frame together with the four ground-truth future frames and their corresponding multi-horizon predictions:

$$\boldsymbol{I}_t \rightarrow \left\{ \hat{\boldsymbol{I}}_{t+1}, \hat{\boldsymbol{I}}_{t+2}, \hat{\boldsymbol{I}}_{t+3}, \hat{\boldsymbol{I}}_{t+4} \right\}. \tag{33}$$

Across the qualitative examples, the predicted representations preserve the global road geometry and capture changes in the position and scale of surrounding vehicles. For example, when the ego vehicle approaches a vehicle ahead, the predicted future representations capture the increasing apparent size of that vehicle while maintaining the surrounding lane structure.

These examples complement the feature-space evaluation by showing that the predicted representations retain temporally meaningful scene structure across the four horizons.



Figure 3: Qualitative DeltaTok finetuning example 2.

C.5 Integration into the Driving Policy

We next evaluate whether the richer multi-horizon representation improves the closed-loop VLA policy. In the default model, the future-fusion module uses a single predicted near-future representation:

$$\boldsymbol{H}_{\text{fut}} = \text{Proj}_{\text{fut}} \left(\hat{\boldsymbol{F}}_{t+1} \right). \tag{34}$$

Table 13: Closed-loop comparison between single-step and multi-horizon future representations on Bench2Drive-220.

Future Representation	**DS** ↑	**Mean Ratio** ↑
Single-step $\boldsymbol{z}_{t+1}$	**88.20** $\pm$ **1.04**	0.0443×
Multi-horizon $\{\boldsymbol{z}_{t+1}, \ldots, \boldsymbol{z}_{t+4}\}$	85.86 $\pm$ 0.52	**0.0477**×

For the multi-horizon variant, the future branch instead provides a set of four temporal representations:

$$\mathcal{H}_{\text{fut}} = \left\{ \boldsymbol{H}^{\text{fut}}_{t+1}, \boldsymbol{H}^{\text{fut}}_{t+2}, \boldsymbol{H}^{\text{fut}}_{t+3}, \boldsymbol{H}^{\text{fut}}_{t+4} \right\}. \tag{35}$$

These representations provide the future-fusion module with information covering several temporal scales rather than only the immediate next state. We then evaluate the resulting policy in closed loop on Bench2Drive-220.

C.6 Closed-Loop Results

Table 13 compares the single-step and multi-horizon variants in closed-loop driving.

Despite producing meaningful future representations at all four horizons, the multi-horizon variant does not improve closed-loop driving. Its Driving Score decreases from 88.20 ± 1.04 with single-step prediction to 85.86 ± 0.52 with multi-horizon prediction. The multi-horizon variant nevertheless obtains a slightly higher measured throughput ratio, $0.0477\times$ compared with $0.0443\times$.

This result reveals an important distinction between *future prediction quality* and *future information utility for control*. The feature-reconstruction results demonstrate that the model can represent multiple future horizons, but providing more predicted future information does not automatically produce better closed-loop decisions.

C.7 Discussion

The results suggest that the most useful temporal horizon for the driving policy is not necessarily the longest one. Closed-loop control requires selecting an action for the current timestep, and the immediate future is therefore closely coupled to the decision that must be executed next. A single near-future representation can provide information about imminent scene evolution while remaining compact and directly relevant to the current control decision.

In contrast, multiple future representations contain information at different temporal scales. Although each prediction can be individually meaningful, the driving policy must determine which horizon is relevant to the current action. Longer-horizon features may also contain information that is less directly actionable or more uncertain from the perspective of the current state. Consequently, increasing the number of future tokens does not guarantee an improvement in planning.

Importantly, the closed-loop degradation should not be interpreted as a failure of the multi-horizon DeltaTok representation itself. The multi-horizon predictor remains above the static feature baseline at every evaluated horizon, with the largest improvement occurring at the longest horizon. Instead, the experiment indicates that the current VLA fusion mechanism benefits more from a compact immediate-future representation than from simultaneously exposing the policy to several temporal horizons.

These results motivate our final design choice:

$$\boxed{\text{Immediate Future} \;>\; \text{Multiple Future Horizons} \quad \text{for the current closed-loop fusion design.}} \tag{36}$$

Accordingly, the final ProActiveDrive configuration uses single-step future prediction. This choice provides the strongest closed-loop Driving Score while retaining a compact future-aware representation.

D DATA ENHANCEMENT

The quality of language supervision is particularly important for vision-language-action (VLA) driving models because the textual annotation is expected to describe the driving decision represented by the expert trajectory. In our initial training data, decision-critical frames were associated with a language target of the form

$$\langle \text{slow} \rangle \ \{\text{reasoning}\} \ \text{Waypoints} :, \tag{37}$$

where the reasoning component was generated using a template-based commentary system. Although these templates provide inexpensive language supervision at scale, we observed that they can describe the wrong cause of an expert action in interaction-heavy scenarios. We therefore introduce a data-enhancement pipeline that replaces generic template-based reasoning with scene-specific grounded reasoning while preserving the original images and expert waypoint supervision.

D.1 LIMITATIONS OF TEMPLATE-BASED DRIVING COMMENTARY

The original commentary generator relies on predefined rules to convert driving states into textual descriptions. This works reasonably well for simple situations such as following a genuine leading vehicle or stopping at an explicitly detected traffic light. However, it becomes unreliable when the correct behavior depends on the semantic role of surrounding agents.

A recurring failure mode is that nearby vehicles are interpreted as leading vehicles to follow, even when they are crossing the ego trajectory, approaching from the opposite direction, or have priority at an intersection. Consequently, the generated language can disagree with the expert trajectory. For example, an expert trajectory may slow down to yield to an oncoming vehicle while the associated commentary instructs the model to accelerate and follow a vehicle ahead.

More generally, let the original training sample at timestep t be

$$\mathcal{X}_t = (\boldsymbol{I}_t, c_t, y_t, \tau_t^*)\,, \tag{38}$$

where $\boldsymbol{I}_t$ denotes the front-view image, c_t the navigation command, y_t the original language annotation, and τ_t^* the expert trajectory. Ideally, the language target should describe the driving decision encoded by τ_t^*. With generic templates, however, it is possible to obtain

$$\text{Intent}(y_t) \neq \text{Intent}(\tau_t^*), \tag{39}$$

creating inconsistent supervision between the language and action targets.

An audit of the original annotations confirmed this issue. In yield-related situations, only 33% of the inspected template-based labels expressed the correct yield intent. This motivated us to improve the *quality* of the existing language annotations rather than simply increase the quantity of reasoning data.

D.2 GROUNDED REASONING ANNOTATION

We regenerate the reasoning annotations using Qwen3-VL-2B. Importantly, the annotation model is not asked to infer the complete driving situation from the front-view image alone. During offline annotation, it additionally receives privileged simulator information describing the actual state of the scene.

For each decision frame, the grounding information contains object-level information for nearby traffic participants, including their semantic role, distance from the ego vehicle, and speed. We additionally provide traffic-light state, stop-sign state, and information about the object responsible for the ego vehicle slowing down when such an object is available.

We denote the privileged object information by

$$\mathcal{O}_t = \left\{ \left(r_t^{(j)}, d_t^{(j)}, v_t^{(j)} \right) \right\}_{j=1}^{N_t}, \tag{40}$$

where N_t is the number of relevant nearby agents, $r_t^{(j)}$ denotes the semantic role of agent j, $d_t^{(j)}$ its distance from the ego vehicle, and $v_t^{(j)}$ its speed. Semantic roles distinguish, for example, a leading vehicle from an oncoming or crossing vehicle.

Traffic-rule and ego-state information is represented as

$$\mathcal{S}_t = \left(s_t^{\text{light}}, s_t^{\text{stop}}, o_t^{\text{slow}}, v_t^{\text{ego}} \right), \tag{41}$$

where s_t^{light} denotes the traffic-light state, s_t^{stop} denotes the stop-sign state, o_t^{slow} identifies the object associated with the ego vehicle's slowing behavior when available, and v_t^{ego} denotes the ego speed.

The grounded annotation is then generated as

$$\tilde{y}_t = G_{\text{VLM}} \left(\boldsymbol{I}_t, c_t, \mathcal{O}_t, \mathcal{S}_t \right), \tag{42}$$

where G_{VLM} denotes the Qwen3-VL-2B annotation process. The objective of the annotation model is to identify the relevant interaction in the current scene and generate reasoning consistent with the actual expert behavior.

The resulting enhanced sample becomes

$$\tilde{\mathcal{X}}_t = \left(\boldsymbol{I}_t, c_t, \tilde{y}_t, \tau_t^* \right). \tag{43}$$

A critical property of this procedure is that the visual observations and expert trajectories are never modified:

$$\boldsymbol{I}_t^{\text{new}} = \boldsymbol{I}_t, \qquad \tau_t^{*,\text{new}} = \tau_t^*. \tag{44}$$

Thus, the enhancement changes only the language supervision. It does not introduce synthetic trajectories or modify the expert driving policy.

D.3 Privileged Spatial Grounding

The main purpose of privileged perception is to remove ambiguity about *which* object explains the expert action. This distinction is particularly important in dense scenes containing several vehicles.

Consider a frame containing a nearby vehicle in front of the ego vehicle and an oncoming vehicle with priority. A template based primarily on proximity can describe the situation as car following. In contrast, privileged object information explicitly identifies the oncoming vehicle and its spatial relation to the ego trajectory. The annotation model can therefore associate the expert deceleration with yielding rather than following.

Conceptually, the annotation process performs

$$\underbrace{(\boldsymbol{I}_t, \mathcal{O}_t, \mathcal{S}_t)}_{\text{scene-specific evidence}} \longrightarrow \underbrace{\text{RelevantObject}_t}_{\text{interaction grounding}} \longrightarrow \underbrace{\text{DrivingIntent}_t}_{\text{yield, follow, stop, avoid, ...}} \longrightarrow \underbrace{\tilde{y}_t}_{\text{grounded reasoning}}. \tag{45}$$

This procedure allows the annotation to distinguish actions that can appear similar at the trajectory level but arise from different causes. Slowing down behind a leading vehicle, yielding to an oncoming vehicle, stopping for a red light, and slowing for a pedestrian can all produce similar short-term speed changes, but they correspond to different driving decisions. The enhanced annotation explicitly captures this distinction.

D.4 Scenario-Specific Enhancement

We focus the annotation enhancement on decision-critical scenario families for which the original templates are most likely to produce ambiguous or incorrect explanations. These include yield and give-way interactions, oncoming vehicles with priority, obstacle and construction scenarios, pedestrian interactions, traffic lights, stop signs, and leading-vehicle situations.

Table 14 summarizes representative failure modes observed during the annotation audit.

The final case is important: the goal of the enhancement is not to replace every existing annotation with a different behavior. When the original template correctly describes the scene, the grounded annotation should preserve the same driving intent. The enhancement therefore acts as a correction mechanism rather than systematically producing more complex or more conservative reasoning.

Table 14: Representative differences between template-based commentary and grounded reasoning. The table summarizes the annotation behavior; expert waypoints remain unchanged in all cases.

Scenario	Template-based interpretation	Grounded interpretation
Red traffic light	Generic deceleration due to the traffic light.	Identifies the red-light state explicitly and associates it with a complete stop decision.
Construction / hazard	Encourages acceleration or a lane change around the construction area.	Uses nearby-agent information to identify an interacting vehicle and reasons that yielding should precede the maneuver.
Oncoming vehicle	Treats a nearby vehicle as a leading vehicle and encourages following.	Identifies the vehicle as oncoming, reasons about collision risk and priority, and produces a yield decision.
Pedestrian interaction	Describes only the closest vehicle and ignores other relevant scene participants.	Considers both vehicle interactions and the pedestrian state before selecting the relevant cause of the action.
Leading vehicle	Follows or remains stopped behind the leading vehicle.	Preserves the following interpretation when privileged perception confirms that the nearby vehicle is genuinely leading the ego vehicle.

D.5 Two-Round Annotation Procedure

The data enhancement was performed in two rounds. The first round focused primarily on yield- and obstacle-related situations, where the initial audit revealed the largest mismatch between template commentary and expert behavior. Approximately 70K grounded annotations were generated in this stage.

The second round expanded the coverage by approximately 100K annotations, including additional traffic-sign and leading-vehicle situations. Across both rounds, the final enhanced dataset contains

$$N_{\text{grounded}} = 170{,}241 \tag{46}$$

grounded reasoning annotations.

The two rounds also revealed an important property of reasoning-data augmentation: increasing the amount of CoT supervision is not necessarily equivalent to improving its usefulness. The largest behavioral gains were observed when the enhancement corrected annotations whose original driving intent was wrong, particularly yield and hazard interactions. Additional annotations for scenarios where the original templates were already reliable produced substantially smaller improvements. This motivates targeting annotation errors rather than indiscriminately generating more reasoning traces.

D.6 Annotation Quality Control

Automatically generated reasoning can itself introduce annotation bias. We therefore audit the generated annotations against privileged simulator state before using them for training.

An initial annotation batch revealed a strong tendency toward over-yielding. Approximately 79% of the inspected outputs exhibited over-yield behavior, and traffic-light situations were not consistently mapped to the appropriate stop decision. This occurred because visual reasoning alone could encourage the annotation model to assign excessive importance to surrounding agents.

We address this failure mode in two ways. First, explicit traffic-light and stop-sign states are provided to the annotation model. Second, the annotation prompt is adjusted to discourage unnecessary yielding when no relevant interaction requires it. After this correction, the measured over-yield rate decreases from 79% to 20%, while red-light annotations achieve 94% consistency with the expected stop behavior.

The resulting quality-control process can be summarized as

$$\text{Generate} \rightarrow \text{Audit} \rightarrow \text{Identify Bias} \rightarrow \text{Add Grounding} \rightarrow \text{Regenerate}. \tag{47}$$

Table 15: Quality analysis of the grounded reasoning annotations.

Metric	Before	After	Change
Correct yield intent	33%	**88%**	**+55 pp**
Over-yield rate	79%	**20%**	**−59 pp**
Object grounding	–	**90%**	–
Red-light → stop consistency	–	**94%**	–

This validation step is essential because privileged grounding alone does not guarantee a useful annotation. The generated explanation must identify the correct object *and* associate it with the appropriate driving decision.

D.7 Annotation Quality Analysis

We quantitatively evaluate whether the enhanced labels are better grounded in the simulator state. Table 15 summarizes the main annotation-quality statistics.

Correct yield intent increases from 33% to 88%, demonstrating that the enhanced labels are substantially more consistent with the interaction represented by the expert trajectory. Moreover, 90% of the grounded CoTs refer to an object that actually exists in the corresponding scene. This is particularly important for preventing hallucinated interaction explanations.

The reduction in over-yielding from 79% to 20% further shows that grounding must be accompanied by explicit quality control. Providing traffic-rule state and refining the annotation instructions allows the VLM to distinguish situations that genuinely require yielding from those in which the ego vehicle should continue, follow, or stop for another reason.

D.8 Effect of Annotation Enhancement on Driving

The effect of the enhanced annotations is most visible in scenario families where the original commentary expressed the wrong driving intent. In the first annotation round, which primarily targeted yield and obstacle situations, grounded reasoning improves Overtaking success by 11.1 percentage points and Give-Way success by 10.0 percentage points. The second annotation round, which expanded coverage to traffic-sign and leading-vehicle situations, yields improvements of 5.4 points in Emergency Braking and 2.2 points in Traffic Sign handling relative to the corresponding annotation stage.

At the aggregate level, the template-based model obtains a Driving Score of 87.52 and a Success Rate of 70.0% on Bench2Drive-220. The first grounded annotation round reaches 88.93 DS and 73.2% SR. The larger annotation set remains competitive, but the additional data does not provide the same proportional improvement.

These observations reinforce the central motivation behind our enhancement: the benefit comes primarily from correcting semantically incorrect supervision rather than simply increasing the number of generated reasoning traces. In particular, replacing an incorrect "follow" interpretation with a grounded "yield" interpretation changes the semantic supervision associated with the same expert trajectory.

D.9 Discussion

The data-enhancement pipeline separates *action supervision* from *reasoning supervision*. Expert waypoints remain the authoritative action target, while privileged simulator information is used only during offline annotation to improve the textual explanation associated with that action. No privileged object boxes, traffic-state metadata, or future expert information introduced for annotation is required as an additional input to the deployed policy.

This distinction is important for interpreting the role of grounded CoT. The objective is not to teach the policy to reproduce privileged simulator state at inference time, nor to generate longer reasoning

traces. Instead, privileged information serves as an annotation tool that identifies the actual cause of the expert decision and converts it into a more reliable language target.

Our experiments further indicate that annotation quality is more important than annotation volume. Grounded reasoning provides its largest benefit when it resolves a genuine semantic inconsistency between the original commentary and expert action. Conversely, adding more reasoning examples to scenarios already described correctly by the template provides a smaller benefit and can introduce new biases if the generated annotations are not carefully audited.

Code and Model Release. The source code, training and evaluation configurations, and trained model weights will be publicly released upon acceptance to facilitate reproducibility and future research.