BLUE: Toward Better Language Use in Efficient Vision-Language-Action Models for Autonomous Driving

George Ling, Lijin Yang, Hao Yang, Zhongzhan Huang†
Bosch Research †Correspondence to: hru4sgh@bosch.com
Code Hugging Face (Data/Logs/Ckpts) Homepage

Abstract

We present **BLUE**, a minimal method for better language use in vision-language-action (VLA) models for autonomous driving (AD). Through extensive analysis, we reveal that language matters on only a small fraction of routes, but on those routes it can greatly improve or degrade performance. Generating language at every frame is therefore inefficient, since most computation is spent on frames that do not benefit from language. We further show that pretrained VLA hidden states potentially already encode whether language will benefit a given frame, even though scene complexity and kinematic features alone struggle to predict this. Based on this finding, BLUE trains a lightweight gate on frozen VLA hidden states to decide per frame whether to activate language generation or predict actions directly, without modifying the backbone or requiring additional human annotation. With just a 0.11M-parameter gate, BLUE sets a new state of the art on both benchmarks, achieving 76.2% success rate on Bench2Drive and 36 driving score on Longest6 v2, while delivering 2.54× inference speedup and 8.9% success rate improvement over the backbone. BLUE provides a practical path toward efficient language-augmented AD, showing that VLA models can retain the benefits of language at a fraction of the cost. **Our code, data, logs and checkpoints are fully available on Github**.

1 Introduction

Recent vision-language-action (VLA) models for autonomous driving typically generate natural language to reason about the scene before predicting driving actions (Renz et al., 2025; Yang et al., 2026a). However, the impact of generated language on closed-loop driving has rarely been systematically quantified. First, we conduct ∼2000 GPU hours of closed-loop analysis on full Bench2Drive (Jia et al., 2024) using SimLingo (Renz et al., 2025), a representative VLA driving model, running each route through repeated experiments and categorizing driving outcomes via statistical tests. As Figure 1 shows, the generated language statistically improves driving on only 14.5% of routes, actively degrades it on 23.6%, and has no clear effect on the remaining majority. Yet current many VLA driving models (Renz et al., 2025; Yang et al., 2026a; Gao et al., 2026) usually generate language at every frame by default, wasting computation on frames that do not benefit and compromising both driving performance and inference efficiency. See more analysis on extra settings in Appendix D.2.

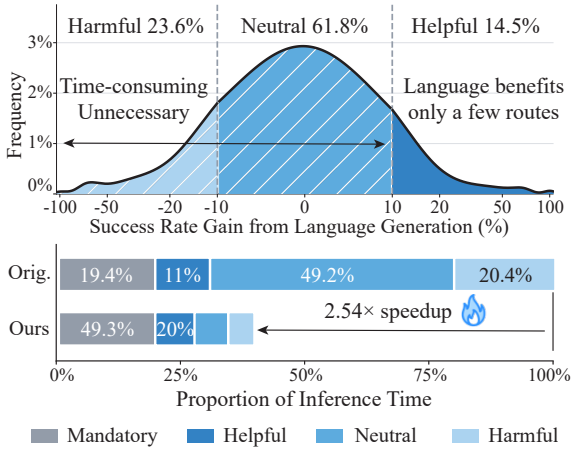

Figure 1: Top: Distribution of Bench2Drive routes by language impact. Bottom: Inference time breakdown comparing the original VLA and BLUE, which reduces unnecessary language generation for 2.54× speedup. Extended details and results are in Appendix D.2.

Since language significantly helps or hurts driving performance on only a minority of routes at substantial inference cost, an intuitive strategy is to detect per frame whether language generation is needed and skip it otherwise, thereby improving both driving performance and inference speed. Fortunately, we find that pretrained VLA hidden states potentially already encode this language-utility signal, even though scene complexity and kinematic features alone struggle to predict it. In Section 3, we leverage this finding and propose BLUE, a minimal method that trains a lightweight gate on frozen

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 43156–43185
October 24-29, 2026 ©2026 Association for Computational Linguistics

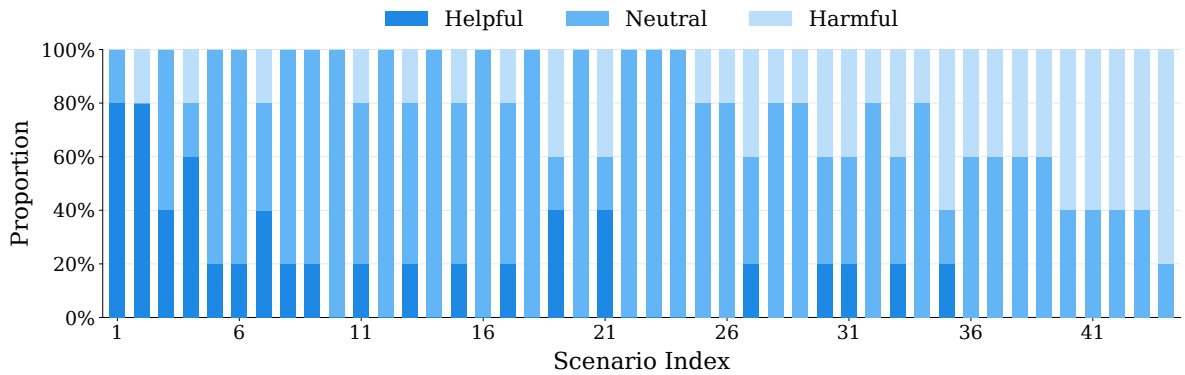

Figure 2: Per-scenario distribution of language effects across all 44 Bench2Drive scenario categories. Each bar represents one scenario, showing the proportion of routes where language generation is helpful, neutral, or harmful to driving success. Extended details and results under additional settings are in Appendix D.2.

hidden states to decide per frame whether to activate language generation or predict actions directly. BLUE requires no backbone modification and no additional human annotation, as training labels are naturally derived from routine driving evaluation. In Section 4, we show that with just a 0.11M-parameter gate, BLUE sets new state of the art on two benchmarks, achieving 76.2% success rate on the multi-scenario Bench2Drive and 36 driving score on the long-horizon Longest6 v2, while delivering 2.54× inference speedup and a 8.9% success rate improvement over the VLA backbone. We discuss related work in Appendix B and summarize our contributions as follows:

- We provide a systematic analysis of when language helps and when it hurts driving, showing that on-demand language use can improve both driving performance and inference speed.
- We reveal that pretrained VLA hidden states potentially already encode the language-utility signal. With just a 0.11M-parameter gate on frozen hidden states, BLUE achieves state-of-the-art performance on Bench2Drive and Longest6 v2, while delivering 2.54× inference speedup.

2 Observations and Motivations

We analyze how generated language affects driving performance across different scenarios and quantify the potential gains from selective language use.

Setup. We study SimLingo (Renz et al., 2025), a VLA driving model that generates natural language reasoning before predicting actions. By skipping language generation, the model can also predict actions directly from its internal representations. We evaluate both configurations on all 44 scenario categories of Bench2Drive (Jia et al., 2024), running repeated experiments per route. Figure 2 shows the per-scenario results. We provide analysis under additional settings in Appendix D.2.

Language Does Not Always Help. As shown in Figure 2, language generation only matters on a small fraction of routes, but where it matters, the impact on driving performance is substantial, either improving or degrading it by a large margin. On the majority of routes, language has no measurable effect yet still incurs full generation cost. Intuitively, if we can detect when language is needed and skip it otherwise, the model could achieve both better driving performance and faster inference.

Room for Improvement. To quantify the potential gains, we construct a route-level oracle that picks the better-performing configuration for each route. Even with this coarse-grained selection, the oracle already reaches 78.4% success rate, revealing more than 10% room for improvement over the default VLA. Since the oracle only makes a single choice per route while finer per-frame selection could further improve performance, this estimate is conservative. The large gap confirms that a lightweight selection mechanism can unlock substantial performance gains without any model retraining, motivating the gate design in Section 3.

3 Method

In this section, we present BLUE, which uses pretrained VLA hidden states to predict per frame whether to activate language generation. Figure 3 shows the overall framework.

Hidden States Encode Language Necessity. To predict when language generation is needed, we look for a signal within the model itself. We

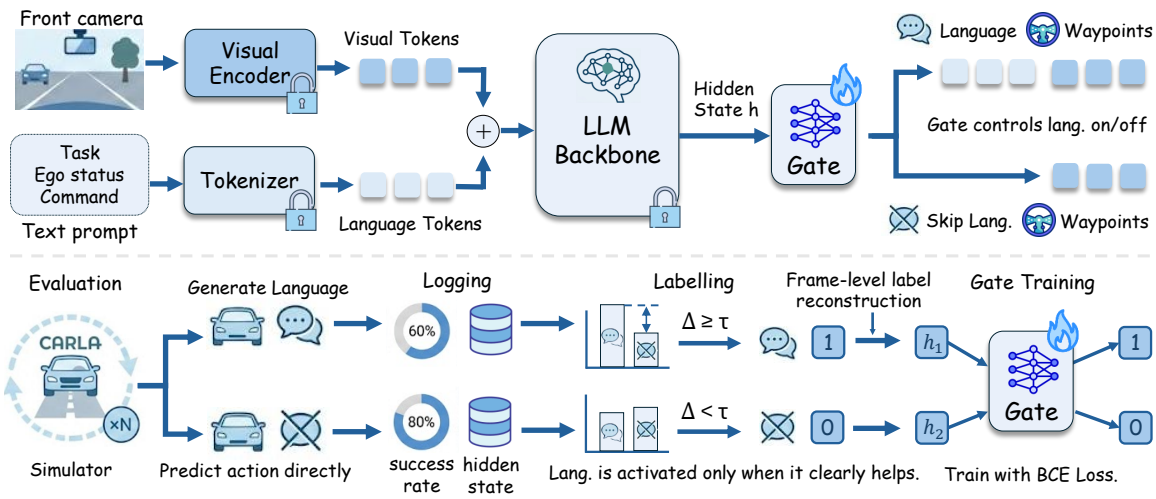

Figure 3: Overview of BLUE. **Top**: a lightweight gate receives the hidden state from the frozen VLA backbone and decides per frame whether to activate language generation or directly output waypoints. **Bottom**: gate training pipeline. Labels are derived from route-level success rate comparisons and refined via frame-level reconstruction. The visual encoder and LLM backbone remain frozen 🔒, while only a lightweight MLP gate is trainable 🔥.

find that a simple logistic regression trained on the VLA's last-layer hidden states can distinguish frames where language helps from those where it does not, without relying on any external features. This shows that the pretrained hidden states potentially already encode a language-utility signal, providing a basis for building a lightweight gate.

Data Collection and Labeling. We run both language mode and direct action mode on training routes through repeated experiments, collecting the last-layer hidden state $\mathbf{h} \in \mathbb{R}^d$ at the final token position for each frame. Details of data splits are provided in Appendix C.1, and additional labeling details are provided in Appendix C.4. This position corresponds to the model's representation right before language or waypoint generation begins, and is shared by both modes to ensure feature alignment. See more details of data splits and labeling in Appendix C.4.2. Each route r is first assigned a binary label based on the success rate gap:

$$y_r = \mathbb{1}\Big[\frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}}\big(\mathrm{SR}^{(r,s)}_{\text{lang}} - \mathrm{SR}^{(r,s)}_{\text{direct}}\big) > \tau\Big], \quad (1)$$

where $\mathcal{S}$ is the set of random seeds, SR denotes success rate, and $\tau{=}10\%$ is a margin threshold. This design defaults to the faster direct action mode unless language shows a clear advantage.

We construct training labels at two granularities. In route-level labeling, every frame on route r shares the same label y_r. In frame-level labeling, we further refine routes that benefit from language ($y_r{=}1$) by marking only the critical regions $\mathcal{C}_r$ where language mode and direct action mode differ the most. We identify these regions from spatial patterns of behavior divergence. See Appendix C.4.2 for details. The frame-level label is:

$$y_{r,t} = \mathbb{1}\big[\Delta\overline{\mathrm{SR}}_r > \tau\big] \cdot \mathbb{1}\big[\mathbf{x}_t \in \mathcal{C}_r\big], \quad (2)$$

where $\Delta\overline{\mathrm{SR}}_r{=}\frac{1}{|\mathcal{S}|}\sum_s(\mathrm{SR}^{(r,s)}_{\text{lang}}{-}\mathrm{SR}^{(r,s)}_{\text{direct}})$ is the cross-seed language advantage of route r, and $\mathbf{x}_t{\in}\mathbb{R}^2$ is the spatial coordinate of frame t. The first indicator selects language-beneficial routes, while the second restricts positive labels to critical segments within those routes. We mix samples from both labeling granularities during training so that the gate learns both route-level preference and frame-level activation. To address temporal redundancy (e.g., near-identical features when the vehicle is stopped), we group consecutive frames with cosine similarity above 0.99 into redundant segments and downsample each segment of length L to $\max(2, \lceil\sqrt{L}\rceil)$ representative samples.

Gate Design. The gate is a single-hidden-layer MLP with negligible parameter count, trained with binary cross-entropy loss. This small design is sufficient for mapping pretrained hidden states to a binary gating decision, while reducing overfitting risk and keeping inference overhead negligible.

On-demand language use. The trained gate is integrated into the VLA inference pipeline with negligible overhead. At each frame, the model computes $\mathbf{h}$ through a forward pass shared by both modes. The gate outputs a score $p(\mathbf{h})$: if it exceeds

Method	Details					Metrics	
	Expert	Camera	LiDAR	Labels	T-Param.	SR (%) ↑	DS ↑
UniAD-Base (Hu et al., 2023b)	Think2Drive	6×	×	O,M,S	≥ 59 M	16.36	45.81
TF++ (Zimmerlin et al., 2024)	PDM-Lite	1×	✓	O,M,S,D	≥ 39 M	67.27	84.21
MomAD (Song et al., 2025)	Think2Drive	6×	×	O,M	≥ 25 M	18.11	47.91
DriveTrans (Jia et al., 2025)	Think2Drive	6×	×	O,M	≈ 646 M	35.01	63.46
Hydra-NeXt (Li et al., 2025d)	Think2Drive	2×	×	-	≥ 25 M	50.00	73.86
DiffusionDrive (Liao et al., 2025)	-	3×	✓	O,S	≈ 60 M	52.72	77.68
ORION (Fu et al., 2025)	Think2Drive	6×	×	O,M,L	≥ 300 M	54.62	77.74
AutoVLA (Zhou et al., 2026)	PDM-Lite	1×	×	L	≥ 1.5 B	57.73	78.84
SimLingo (Renz et al., 2025)	PDM-Lite	1×	×	L	≥ 300 M	67.27	85.07
HiP-AD (Tang et al., 2025)	Think2Drive	6×	×	O,M,D	≈ 97 M	69.09	86.77
ReCogDrive (Li et al., 2025c)	Think2Drive	6×	×	L	≥ 2 B	45.45	71.36
GeRo (Yasarla et al., 2026)	Think2Drive	6×	×	O,M,L	≥ 3 B	60.10	81.90
DeLL (Du et al., 2026)	PDM-Lite	1×	✓	O,S	≥ 38 M	68.63	86.86
R2SE (Liu et al., 2026a)	PDM-Lite	1×	✓	O,M,S,D	≥ 39 M	69.54	86.28
AutoMoT (Huang et al., 2026)	PDM-Lite	1×	✓	-	≈ 1.6 B	70.00	87.34
BevAD (Holtz et al., 2026)	PDM-Lite	6×	×	O	≥ 25 M	72.73	88.11
CriticVLA (Yang et al., 2026a)	PDM-Lite	1×	×	L	≥ 300 M	73.33	88.02
TakeVLA (Gao et al., 2026)	PDM-Lite	1×	×	L	≥ 300 M	73.73	89.72
BLUE (Ours)	PDM-Lite	1×	×	L	**0.11 M**	**76.18**$_{\pm 0.64}$	**90.58**$_{\pm 0.12}$
Δ vs. SimLingo	-	-	-	-	-	+8.91	+5.51

Table 1: Results on Bench2Drive. BLUE achieves the highest closed-loop success rate (SR) and driving score (DS), with large margins over its SimLingo backbone. T-Param. reports trainable parameters; we use published values (≈) where available and conservative lower bounds (≥) derived from the minimum size of trained components. BLUE trains only a 0.11M gate while keeping the VLA backbone frozen. Notably, BLUE surpasses methods that employ multi-camera setups, LiDAR, or dense auxiliary labels (O: 3D object detection, M: map, S: semantic segmentation, D: depth, L: language), using only front-view camera with language annotations. See more results in Appendix D.1.

a threshold θ, the model proceeds with language generation; otherwise, it produces actions directly. The threshold θ governs how selectively the gate triggers language generation at inference time.

4 Experiments

In this section, we evaluate BLUE on two established closed-loop driving benchmarks and compare against a wide range of published methods.

Setup. We apply BLUE to SimLingo (Renz et al., 2025) with a gate threshold of θ=0.66. The gate uses a hidden dimension of 128 and is trained on approximately 400 routes sampled from SimLingo's training set. We evaluate on two benchmarks: Bench2Drive (Jia et al., 2024), a multi-scenario benchmark covering 220 routes across 44 scenario categories in CARLA (Dosovitskiy et al., 2017), and Longest6 v2 (autonomousvision, 2026), a long-horizon benchmark that evaluates sustained driving quality through driving score, route completion, and infraction score. We compare against 26 published methods spanning end-to-end and VLA approaches. All BLUE results are averaged over 3 random seeds. More details on data splits, benchmarks, baselines, implementation, and additional results are provided in Appendix C and D.

Closed-Loop Results on Bench2Drive. Table 1 presents the main comparison on Bench2Drive. BLUE achieves the highest success rate and driving score among all compared methods, improving both metrics over its backbone SimLingo by a large margin. Notably, BLUE trains only a 0.11M-parameter gate on a frozen backbone, uses a single front-view camera without LiDAR, and requires only language annotations. Despite this minimal setup, it surpasses methods that employ six cameras, LiDAR, dense auxiliary labels, or orders-of-magnitude more trainable parameters. Table 2 further reports the multi-ability breakdown. BLUE ranks second in mean ability score, close to the best method that relies on six cameras, with clear improvements in overtaking and emergency braking.

Method	Details		Multi-Ability Success Rate (%)					
	Camera	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
TF++ (Zimmerlin et al., 2024)	1×	✓	58.75	57.77	83.33	40.00	82.11	64.39
DriveTrans (Jia et al., 2025)	6×	×	17.57	35.00	48.36	40.00	52.10	38.60
Hydra-NeXt (Li et al., 2025d)	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
HiP-AD (Tang et al., 2025)	6×	×	50.00	84.44	83.33	40.00	72.10	65.98
SimLingo (Renz et al., 2025)	1×	×	53.78	67.41	81.67	50.00	77.20	66.01
ReCogDrive (Li et al., 2025c)	6×	×	29.73	20.00	69.09	20.00	71.34	42.03
GeRo (Yasarla et al., 2026)	6×	×	40.06	78.24	87.32	50.00	76.83	66.49
R2SE (Liu et al., 2026a)	1×	✓	53.33	61.25	90.00	50.00	84.21	67.76
DeLL (Du et al., 2026)	1×	✓	61.25	62.22	80.00	60.00	81.05	68.90
TakeVLA (Gao et al., 2026)	1×	×	63.64	64.44	91.67	50.00	85.48	71.05
CriticVLA (Yang et al., 2026a)	1×	×	61.28	76.30	88.33	50.00	81.06	71.39
BevAD (Holtz et al., 2026)	6×	×	71.67	74.07	75.56	76.67	75.44	74.68
BLUE (ours)	1×	×	61.44±1.33	80.00±1.81	93.27±1.33	50.00±0.00	84.74±0.00	73.89±0.14
Δ vs. SimLingo	-	-	+7.66	+12.59	+11.60	+0.00	+7.54	+7.88

Table 2: Multi-ability results on Bench2Drive. Mean denotes the average success rate over the five driving skills. While using only a single front-view camera and no LiDAR, BLUE achieves the second-best mean result and remains close to the best method, which uses six cameras. See additional results in Appendix D.

Method	DS ↑	RC ↑	IS ↑	Time ↓
HiP-AD (Tang et al., 2025)	7	56	-	-
TF++ (Zimmerlin et al., 2024)	23	71	-	-
SimLingo (Renz et al., 2025)	22	70	0.38	119h
CriticVLA (Yang et al., 2026a)	34	66	**0.55**	193h
BLUE (ours)	**36**	**84**	0.43	**56h**
Δ vs. SimLingo	+14	+14	+0.05	-63h

Table 3: Closed-loop results on Longest6 v2. DS: driving score, RC: route completion, IS: infraction score. Time: total A100 GPU hours to evaluate all routes.

Method	Speed Ratio ↑	FPS ↑	Latency (ms) ↓
HiP-AD	0.0625	1.25	800.3
SimLingo	0.0358	0.72	1396.6
CriticVLA	0.0146	0.29	3424.7
BLUE (ours)	**0.0910**	**1.82**	**549.5**
Δ vs. SimLingo	+154.2%	+154.2%	-60.7%

Table 4: Inference efficiency comparison among representative driving models. Higher speed ratio and FPS are better, while lower latency is better.

Closed-Loop Results on Longest6 v2. Table 3 presents results on Longest6 v2, a long-horizon benchmark that evaluates sustained driving quality. BLUE achieves the highest driving score and route completion among all compared methods, while requiring substantially fewer GPU hours. The improvement in route completion suggests that our BLUE helps maintain robust driving over long distances, where errors from unnecessary language generation would otherwise compound.

Inference Efficiency. Since the gate skips language generation on most frames, BLUE runs substantially faster than the full language mode. The gate itself adds negligible overhead, as it is a single-hidden-layer MLP applied to the already-computed hidden state. As shown in Table 4, BLUE achieves a 2.54× speedup over SimLingo with the lowest latency among all compared methods. We provide additional efficiency analysis in the next section.

5 Analysis and Ablation Study

We now analyze the language gate from multiple angles: its activation behavior, generalizability to other models, and sensitivity to design choices.

(1) What activation pattern does the gate learn?

We visualize the gate's frame-level decisions across evaluation routes in Figure 4. The gate skips language generation on most frames, yet BLUE still outperforms the VLA backbone by a large margin, as shown in Section 4. When it does activate language, the activations form contiguous segments rather than scattering across isolated frames, suggesting that the gate captures temporally coherent patterns from the hidden states rather than producing noisy frame-level fluctuations.

(2) Can BLUE be applied to other models?

The main experiments use SimLingo as the backbone. To examine whether BLUE transfers to other

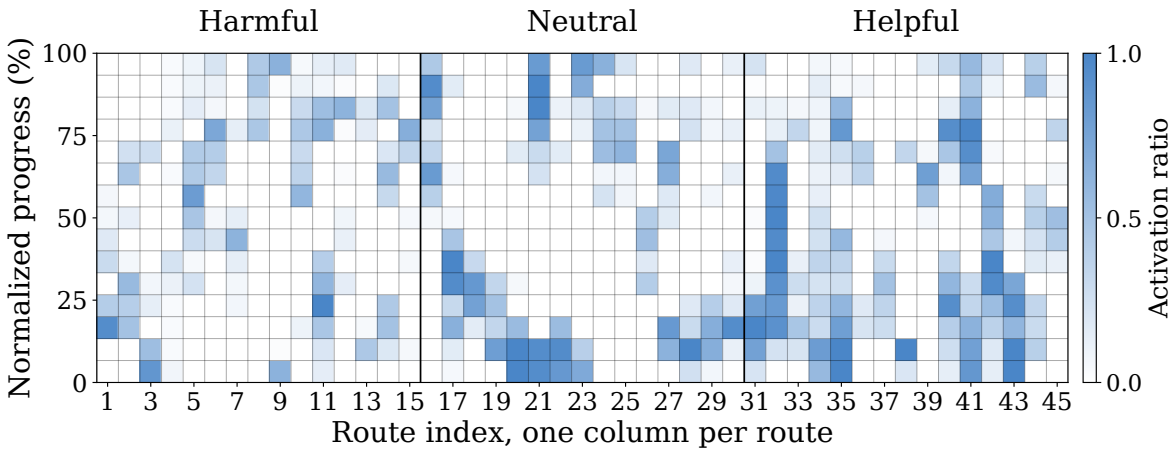

Figure 4: Language activation pattern learned by the gate. Each column is one route, grouped by the route-level language benefit measured in Section 2. The color encodes the fraction of frames where language generation is activated at each progress bin. The gate activates language sparingly and in contiguous segments.

Method	Bench2Drive		Longest6 v2	
	SR (%) ↑	DS ↑	DS ↑	RC ↑
TF++ (Zimmerlin et al., 2024)	67.27	84.21	23	71
HiP-AD (Tang et al., 2025)	69.09	86.77	7	56
SimLingo (Renz et al., 2025)	67.27	85.07	22	70
CriticVLA (Yang et al., 2026a)	73.33	88.02	34	66
BLUE (CriticVLA)	**76.04**	**90.37**	**36.2**	**80.6**
Δ vs. CriticVLA	+2.71	+2.35	+2.2	+14.6

Table 5: BLUE applied to CriticVLA, compared with representative methods on Bench2Drive and Longest. Full comparison results are in Appendix D.

Configuration	SR (%) ↑	DS ↑
SimLingo (Renz et al., 2025)	67.27±2.11	85.07±0.95
SimLingo + CriticVLA gate	71.59±0.96	89.23±0.21
SimLingo + SimLingo gate	**76.18**±0.64	**90.58**±0.12
CriticVLA (Yang et al., 2026a)	73.33±0.27	88.02±0.17
CriticVLA + SimLingo gate	73.11±1.36	88.90±0.53
CriticVLA + CriticVLA gate	**76.04**±0.38	**90.37**±0.14

Table 6: Cross-model transfer of the learned language gate on Bench2Drive. Each transferred gate is evaluated on a model different from the one used for training.

architectures, we apply the same pipeline to CriticVLA (Yang et al., 2026a), a VLA model with a different language integration design. As shown in Table 5, BLUE improves CriticVLA across all metrics on both Bench2Drive and Longest6 v2, with particularly notable gains in route completion on Longest6 v2. The resulting system also surpasses all other listed baselines. This suggests that the language-utility signal in hidden states is not unique to SimLingo, and that the gate mechanism can benefit other language-centric VLA models. Full comparison results are in Appendix D.

(3) Is the gate transferable across models?

Since BLUE trains a separate gate for each model, a natural question is whether a gate learned on one model can be directly reused on another without retraining. We test this by swapping the trained gates between SimLingo and CriticVLA. As shown in Table 6, the matched gate consistently outperforms the transferred gate by a clear margin. This indicates that when language helps is inherently tied to the model itself rather than to the driving scenario. Different models internalize language utility in distinct ways within their hidden states, so each model should train its own gate. Appendix D.2 further analyzes why each model requires its own gate and its retraining cost.

(4) Are rule-based methods sufficient for predicting when language is needed?

We ask whether simple driving signals can replace the learned gate. We design three rule-based gates that activate language when a kinematic feature exceeds a threshold: vehicle speed, acceleration magnitude, or steering angle. See Appendix C.6 for construction details. As shown in Table 7, regardless of which feature or threshold is used, all rule-based gates fall far short of BLUE and offer only marginal gains over single-mode baselines. The random gate performs even worse. This result is expected: kinematic features reflect only the vehicle's current motion and do not provide enough information to predict whether language will help. The VLA's hidden states, which jointly

Gate Method	SR (%) ↑	DS ↑	Lang. (%)
Speed-based gate	70.97±1.21	88.71±0.63	55.50±1.21
Speed-based gate	71.81±0.54	89.91±0.88	30.21±0.91
Acceleration-based gate	70.08±0.43	88.33±0.43	49.12±0.87
Steering-based gate	70.71±0.97	89.38±0.22	7.94±0.02
Complexity-based gate	70.98±0.80	87.12±1.75	53.61±1.08
Complexity-based gate	71.40±2.26	88.02±0.85	17.15±0.58
Random gate	67.42±0.01	86.66±1.18	79.90±0.16
Random gate	70.96±0.71	88.36±0.16	50.07±0.41
Random gate	70.01±1.53	87.10±1.58	20.15±0.32
BLUE (ours)	**76.18**±0.64	**90.58**±0.12	21.44±0.66

Table 7: Comparison of alternative gating strategies on Bench2Drive. Kinematic gates activate language when a motion feature exceeds a threshold; the complexity gate activates language on complex routes. Lang. denotes the percentage of frames with language activation.

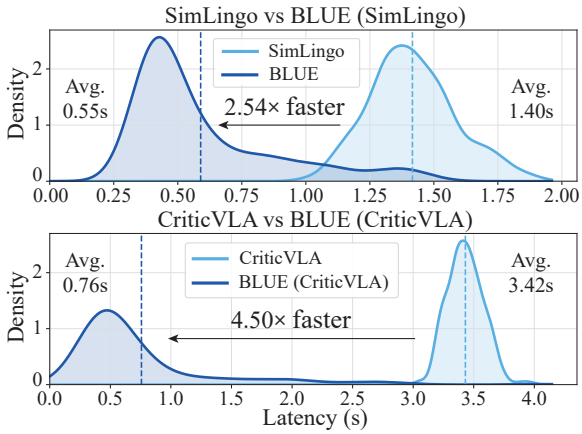

Figure 5: Distribution of mean per-frame latency across Bench2Drive evaluation routes. Dashed lines indicate overall means. BLUE substantially reduces latency for both backbones, delivering 2.54× and 4.50× speedup.

encode perceptual and contextual information, are a much stronger signal for this prediction.

(5) Is scenario complexity sufficient for predicting when language is needed?

An intuitive hypothesis is that language generation is needed in complex scenarios and can be skipped in simple ones. To test this, we compute a composite complexity score for each training route based on structured features including the number of sub-scenarios, weather severity, traffic flow density, and opposing vehicle interactions. Routes above a threshold are labeled complex and activate language generation, while the rest skip it and predict actions directly. See Appendix C.6 for more details. As shown in Table 7, the complexity-based gate performs comparably to the kinematic-based gates and remains far below BLUE. This indicates that whether language generation helps is not determined by scenario complexity alone, and frame-level hidden states capture dynamics that coarse route-level labels cannot provide. We discuss why complexity-based gate fall short in Appendix D.2.

(6) Does BLUE improve inference efficiency?

Beyond driving performance, BLUE also improves inference efficiency. Since the gate skips language generation on most frames and the gate itself is a single-hidden-layer MLP with negligible overhead, BLUE reduces the per-frame latency substantially. Table 4 reports the aggregate statistics: BLUE on SimLingo achieves 2.54× speedup and reduces mean latency from 1.40 s to 0.55 s. The gain extends to CriticVLA, where BLUE achieves 4.50× speedup and lowers the mean latency from 3.42 s to 0.76 s. Figure 5 further shows the per-route latency distributions across all evaluation routes. For both backbones, BLUE shifts the entire distribution toward lower latency. These results confirm that BLUE simultaneously improves both driving quality and inference efficiency.

(7) How sensitive is the gate to the threshold?

The gate threshold θ controls how readily the model activates language generation. We select θ without tuning based on a simple observation: the effect of language on each route falls into three natural categories, helpful, neutral, and harmful, which partition the gate output range $[0, 1]$ into three equal intervals. We place θ at the boundary between neutral and helpful, yielding θ=0.66 directly from this categorization. Figure 6 shows the effect of varying θ across the full range. Language activation ratio decreases monotonically as θ increases, confirming that the gate produces well-calibrated scores. SR peaks near our chosen value, and thresholds from 0.6 to 0.8 all achieve good results. When θ is very low, the model generates language reasoning on nearly every frame before acting, and SR reduces to 66.91%. When θ is very high, the model skips language and produces actions directly at every frame, yielding SR = 69.55%. BLUE at θ=0.66 achieves 76.18% SR, surpassing these two settings by a large margin.

(8) How does training data size affect the gate?

We collect approximately 400 training routes from the SimLingo training set, vary the proportion used from 10% to 100%, and evaluate closed-loop driving on Bench2Drive. Training and evaluation

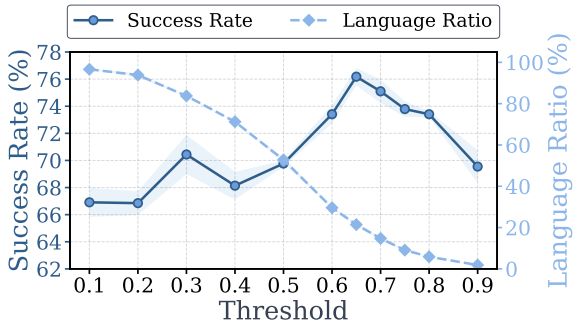

Figure 6: Effect of threshold θ on language activation ratio and success rate. Language usage decreases monotonically with θ, while $\theta \in [0.6, 0.8]$ yields strong SR.

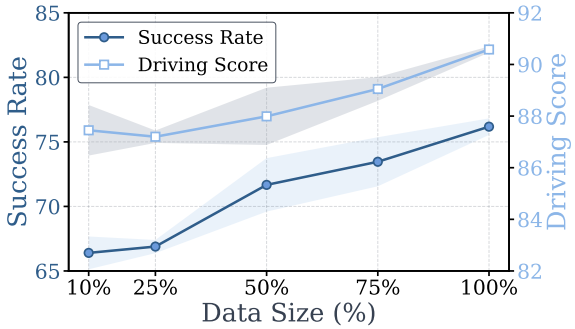

Figure 7: Effect of training data size on gate performance. Both SR and DS improve quickly with more training data and then approach saturation.

Hidden Dim	Dropout	Success Rate ↑	Driving Score ↑
128	No	74.67±0.69	90.25±0.20
128	Yes	**76.18**±0.64	**90.58**±0.12
256	No	74.62±0.33	89.97±0.36
256	Yes	75.19±1.56	89.70±1.15

Table 8: Effect of gate hidden dimension and dropout on Bench2Drive. All settings use the same training data, labels, and inference threshold.

Category	Setting	Helpful	Neutral	Harmful
Original	SimLingo	14.5%	61.8%	23.6%
Granularity	Brief	21.8%	52.7%	25.5%
Language	Chinese	18.6%	58.2%	23.2%
Model	CriticVLA	19.5%	52.3%	28.2%

Table 9: Language usefulness across different settings. Helpful, neutral, and harmful denote the proportion of routes where language helps, has no effect, or hurts.

routes have no overlap. We detail the data splits in Appendix C.1. As shown in Figure 7, both SR and DS improve steadily as training data increases. With only half of the available routes, the gate already surpasses the SimLingo backbone by a clear margin, indicating that the language-utility signal encoded in hidden states is learnable from moderate amounts of data. Performance continues to improve with additional training routes, and variance across seeds decreases, reflecting more stable gate decisions. We use the full training set in all other experiments for best results.

(9) How does gate design affect performance?

We vary the gate hidden dimension and dropout setting while fixing all other factors. As shown in Table 8, all configurations achieve strong performance, and increasing gate capacity does not bring clear improvement. Among all variants, dropout provides a noticeable benefit. We adopt the smaller gate with dropout as the default, since a smaller capacity combined with regularization better prevents overfitting to training routes. This choice is made purely based on the principle of minimizing overfitting risk, not from test-set tuning.

(10) Is the impact of language use consistent across experimental settings?

In Section 2, we observed a performance gap and complementarity between driving with and without language generation on SimLingo. Here we test whether this pattern persists across different experimental settings by varying annotation granularity from normal to brief, switching annotation language from English to Chinese, and replacing the backbone from SimLingo to CriticVLA. As shown in Table 9, the same pattern holds across all settings: language improves driving on only a minority of routes, actively degrades it on another subset, and has no clear effect on the majority. This consistent complementarity across annotation settings and backbones supports applying BLUE to different VLA models. We provide statistical testing details for each setting in Appendix D.2.

(11) Can BLUE handle long-tail scenarios?

To assess generalization to unseen long-tail scenarios, we evaluate BLUE on Fail2Drive (Gerstenecker et al., 2026), a closed-loop benchmark featuring rare cases such as animal crossings and fully blocked roads. As shown in Table 10, BLUE outperforms its SimLingo backbone on every metric in both the in-distribution and generalization settings, while achieving the highest in-distribution success rate among camera-only methods. In the challenging generalization test, BLUE improves DS from 71.7 to 73.9, SR from 55.0% to 59.0%, and HM from 62.2 to 65.6, confirming that its gains over SimLingo persist under long-tail conditions.

Method	RGB	LiDAR	Privileged	Learned	Bench2Drive	Fail2Drive					
					DS ↑	In-Distribution			Generalization		
						DS ↑	SR (%) ↑	HM ↑	DS ↑	SR (%) ↑	HM ↑
TCP (2022)	✓	×	×	✓	59.9	24.7	39.1	30.3	24.5 (-0.8%)	31.4 (-19.7%)	27.5 (-9.1%)
UniAD (2023b)	✓	×	×	✓	45.8	47.5	36.3	41.2	44.0 (-7.4%)	27.6 (-24.0%)	33.9 (-17.6%)
ORION (2025)	✓	×	×	✓	77.8	53.0	52.0	52.5	51.2 (-3.4%)	46.0 (-11.5%)	48.5 (-7.7%)
HiP-AD (2025)	✓	×	×	✓	86.8	74.1	70.7	72.4	67.1 (-9.4%)	56.7 (-19.8%)	61.5 (-15.1%)
SimLingo (2025)	✓	×	×	✓	85.1	82.6	79.3	80.9	71.7 (-13.2%)	55.0 (-30.6%)	62.2 (-23.1%)
AlignDrive (2026)	✓	×	×	✓	89.1	74.7	75.7	75.2	68.6 (-8.1%)	62.7 (-17.2%)	65.5 (-12.9%)
BevAD (2026)	✓	×	×	✓	88.1	87.4	83.3	85.3	82.3 (-5.8%)	68.7 (-17.6%)	74.9 (-12.2%)
BLUE (ours)	✓	×	×	✓	90.6	85.7	84.0	84.8	73.9 (-13.8%)	59.0 (-29.8%)	65.6 (-22.7%)
TF++ (2024)	✓	✓	×	✓	84.2	83.3	78.5	80.8	75.4 (-9.5%)	61.1 (-22.2%)	67.5 (-16.5%)
TFv6 (2026)	✓	✓	×	✓	95.0	90.2	93.3	91.7	79.5 (-11.8%)	70.7 (-24.3%)	74.8 (-18.4%)
BridgeDrive (2025)	✓	✓	×	✓	96.3	91.6	95.0	93.3	81.9 (-10.5%)	75.0 (-21.1%)	78.3 (-16.0%)
PlanT 2.0 (2025)	–	–	✓	✓	92.4	87.8	85.0	86.4	73.3 (-16.5%)	58.0 (-31.8%)	64.8 (-25.0%)
PDMLite-F2D	–	–	✓	×	97.0	95.6	97.0	96.3	94.0 (-1.7%)	95.3 (-1.8%)	94.6 (-1.7%)

Table 10: Results on Fail2Drive. In-Distribution covers standard CARLA scenarios, while Generalization covers out-of-distribution long-tail scenarios. HM is the harmonic mean of DS and SR. Parentheses report the relative change from In-Distribution to Generalization. Our results are averaged over three random seeds.

Method	Lang. Ratio	EP ↑	NC ↑	DAC ↑	DDC ↑	TTC ↑	Comfort ↑	PDMS ↑
ReCogDrive (2025c)	100%	79.08	97.64	92.99	97.80	92.88	**99.99**	84.13
ReCogDrive (2025c)	0%	80.53	98.19	94.14	97.83	94.09	99.98	85.98
BLUE (ReCogDrive)	38.27%	**81.35**	**98.50**	**94.77**	**97.88**	**94.84**	**99.99**	**87.00**

Table 11: Results on NAVSIM. All settings use the same ReCogDrive backbone and differ only in language use. BLUE outperforms both ReCogDrive baselines, demonstrating its generalizability to real-world driving data.

(12) Does BLUE improve real-world driving?

To assess whether BLUE extends beyond simulation, we integrate it with ReCogDrive (Li et al., 2025c) and evaluate it on NAVSIM (Dauner et al., 2024), a benchmark built from real-world driving data. ReCogDrive supports two ways to produce a trajectory. With language generation, it autoregressively generates the trajectory as text and converts it into numerical waypoints for execution. When language generation is skipped, it generates no tokens and instead passes the VLM hidden state to a planner trained by imitation learning on the NAVSIM training set, which predicts the trajectory directly. BLUE uses the frozen VLM hidden state to choose between these two options at each frame. We train the gate on 90% of the NAVSIM training set and use the remaining 10% to select its decision threshold. As shown in Table 11, BLUE outperforms both fixed strategies, showing that selective language use remains effective on real-world driving data. However, evaluation on recorded real-world data is not equivalent to deployment on real vehicles. Such deployment would require closed-loop testing under sensor noise, distribution shifts, and safety-critical conditions, as discussed in Appendix F.

6 Conclusion

We present BLUE, a minimal method for better language use in VLA driving. We reveal that generated language helps in some situations, hurts in others, and is unnecessary most of the time. Based on this observation, BLUE trains a 0.11M-parameter gate on frozen VLA hidden states to decide when to generate language. It achieves 76.2% success rate on Bench2Drive, 36 driving score on Longest6 v2, setting new state of the art while delivering 2.54× inference speedup. Results across three VLA backbones and four benchmarks further demonstrate the broad applicability of BLUE. These results suggest that better language use in VLA driving does not come from generating more language, but from generating language only when it improves driving.

Limitations

Despite its effectiveness, BLUE has two limitations. First, BLUE introduces uneven per-frame latency, as frames that skip language generation run faster than those that generate language. However, this is not unique to BLUE. Language-generating VLA systems inherently exhibit variable per-frame latency because the number of output tokens differs across frames. Since the gate adds negligible overhead, BLUE barely increases the maximum per-frame latency compared with the original VLA, while substantially reducing the average. Second, BLUE requires training a separate gate for each backbone. However, the gate is a lightweight single-layer MLP and the VLA backbone remains entirely frozen, so the adaptation cost is low. Training labels are a natural byproduct of routine driving evaluation and require no additional human annotation or reward engineering. Applying BLUE to a new backbone is therefore considerably cheaper than retraining or modifying the backbone itself.

Acknowledgments

AI assistants were used in this work exclusively for English language polishing and assisting with code implementation. All research ideas, experimental designs, analyses, and scientific conclusions are original contributions of the authors. The authors reviewed all LLM-assisted outputs and take full responsibility for the final content of this paper.

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*.

Daman Arora and Andrea Zanette. 2026. Training language models to reason efficiently. *Advances in Neural Information Processing Systems*, 38:60770–60808.

autonomousvision. 2026. Carla garage: A starter kit for the carla leaderboard 2.0. https://github.com/autonomousvision/carla_garage.

Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. 2020. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631.

Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. 2021. nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*.

Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. 2024a. End-to-end autonomous driving: Challenges and frontiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):10164–10183.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, and 1 others. 2024b. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 24185–24198.

Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. 2022. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *IEEE transactions on pattern analysis and machine intelligence*, 45(11):12878–12895.

Muzhi Dai, Chenxu Yang, and Qingyi Si. 2026. S-grpo: Early exit via reinforcement learning in reasoning models. *Advances in Neural Information Processing Systems*, 38:48178–48204.

Daniel Dauner, Marcel Hallgarten, Andreas Geiger, and Kashyap Chitta. 2023. Parting with misconceptions about learning-based vehicle motion planning. In *Conference on Robot Learning*, pages 1268–1281. PMLR.

Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, and 1 others. 2024. Navsim: Data-driven non-reactive autonomous vehicle simulation and benchmarking. *Advances in Neural Information Processing Systems*, 37:28706–28719.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR.

Jiayuan Du, Yuebing Song, Yiming Zhao, Xianghui Pan, Jiawei Lian, Yuchu Lu, Liuyi Wang, Chengju Liu, and Qijun Chen. 2026. Deconfounded lifelong learning for autonomous driving via dynamic knowledge spaces. *arXiv preprint arXiv:2603.14354*.

Razvan-Gabriel Dumitru, Darius Peteleaza, Vikas Yadav, and Liangming Pan. 2025. Conciserl: Conciseness-guided reinforcement learning for efficient reasoning models. *arXiv preprint arXiv:2505.17250*, 4.

Haoyu Fu, Diankun Zhang, Zongchuang Zhao, Jianfeng Cui, Dingkang Liang, Chong Zhang, Dingyuan

Zhang, Hongwei Xie, Bing Wang, and Xiang Bai. 2025. Orion: A holistic end-to-end autonomous driving framework by vision-language instructed action generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 24823–24834.

Shenyuan Gao, Jiazhi Yang, Li Chen, Kashyap Chitta, Yihang Qiu, Andreas Geiger, Jun Zhang, and Hongyang Li. 2024. Vista: A generalizable driving world model with high fidelity and versatile controllability. *Advances in Neural Information Processing Systems*, 37:91560–91596.

Yinfeng Gao, Deqing Liu, Qichao Zhang, Yupeng Zheng, Haochen Tian, Guang Li, Hangjun Ye, Long Chen, Da-Wei Ding, and Dongbin Zhao. 2026. Learning from mistakes: Post-training for driving vla with takeover data. *arXiv preprint arXiv:2603.14972*.

Simon Gerstenecker, Andreas Geiger, and Katrin Renz. 2025. Plant 2.0: Exposing biases and structural flaws in closed-loop driving. *arXiv preprint arXiv:2511.07292*.

Simon Gerstenecker, Andreas Geiger, and Katrin Renz. 2026. Fail2drive: Benchmarking closed-loop driving generalization. *arXiv preprint arXiv:2604.08535*.

Yi Gu, Yan Wang, Yuxiao Chen, Yurong You, Wenjie Luo, Yue Wang, Wenhao Ding, Boyi Li, Heng Yang, Boris Ivanovic, and 1 others. 2026. Accelerating structured chain-of-thought in autonomous vehicles. *arXiv preprint arXiv:2602.02864*.

Cole Gulino, Justin Fu, Wenjie Luo, George Tucker, Eli Bronstein, Yiren Lu, Jean Harb, Xinlei Pan, Yan Wang, Xiangyu Chen, and 1 others. 2023. Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. *Advances in Neural Information Processing Systems*, 36:7730–7742.

Shadi Hamdan, Chonghao Sima, Zetong Yang, Hongyang Li, and Fatma Guney. 2025. Eta: Efficiency through thinking ahead, a dual approach to self-driving with large models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 26529–26538.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. Token-budget-aware llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855.

David Holtz, Niklas Hanselmann, Simon Doll, Marius Cordts, and Bernt Schiele. 2026. What matters for scalable and robust learning in end-to-end driving planners? *arXiv preprint arXiv:2603.15185*.

Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. 2023a. Gaia-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*.

Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, and 1 others. 2023b. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 17853–17862.

Wenhui Huang, Songyan Zhang, Qihang Huang, Zhidong Wang, Zhiqi Mao, Collister Chua, Zhan Chen, Long Chen, and Chen Lv. 2026. Automot: A unified vision-language-action model with asynchronous mixture-of-transformers for end-to-end autonomous driving. *arXiv preprint arXiv:2603.14851*.

Zhongzhan Huang, Guoming Ling, Yupei Lin, Yandong Chen, Shanshan Zhong, Hefeng Wu, and Liang Lin. 2025a. Routereval: A comprehensive benchmark for routing llms to explore model-level scaling up in llms. *arXiv preprint arXiv:2503.10657*.

Zhongzhan Huang, Shanshan Zhong, Pan Zhou, Shanghua Gao, Marinka Zitnik, and Liang Lin. 2025b. A causality-aware paradigm for evaluating creativity of multimodal large language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Jyh-Jing Hwang, Runsheng Xu, Hubert Lin, Wei-Chih Hung, Jingwei Ji, Kristy Choi, Di Huang, Tong He, Paul Covington, Benjamin Sapp, and 1 others. 2024. Emma: End-to-end multimodal model for autonomous driving. *arXiv preprint arXiv:2410.23262*.

Xiaosong Jia, Yulu Gao, Li Chen, Junchi Yan, Patrick Langechuan Liu, and Hongyang Li. 2023a. Driveadapter: Breaking the coupling barrier of perception and planning in end-to-end autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7953–7963.

Xiaosong Jia, Penghao Wu, Li Chen, Jiangwei Xie, Conghui He, Junchi Yan, and Hongyang Li. 2023b. Think twice before driving: Towards scalable decoders for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21983–21994.

Xiaosong Jia, Zhenjie Yang, Qifeng Li, Zhiyuan Zhang, and Junchi Yan. 2024. Bench2drive: Towards multi-ability benchmarking of closed-loop end-to-end autonomous driving. *Advances in Neural Information Processing Systems*, 37:819–844.

Xiaosong Jia, Junqi You, Zhiyuan Zhang, and Junchi Yan. 2025. Drivetransformer: Unified transformer for scalable end-to-end autonomous driving. *arXiv preprint arXiv:2503.07656*.

Bo Jiang, Shaoyu Chen, Bencheng Liao, Xingyu Zhang, Wei Yin, Qian Zhang, Chang Huang, Wenyu Liu, and Xinggang Wang. 2024. Senna: Bridging large vision-language models and end-to-end autonomous driving. *arXiv preprint arXiv:2410.22313*.

Bo Jiang, Shaoyu Chen, Qing Xu, Bencheng Liao, Jiajie Chen, Helong Zhou, Qian Zhang, Wenyu Liu, Chang

Huang, and Xinggang Wang. 2023. Vad: Vectorized scene representation for efficient autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8340–8350.

Zhensheng Jin, Xinze Li, Yifan Ji, Chunyi Peng, Zhenghao Liu, Qi Shi, Yukun Yan, Shuo Wang, Furong Peng, and Ge Yu. 2025. Recut: Balancing reasoning length and accuracy in llms via stepwise trails and preference optimization. *arXiv preprint arXiv:2506.10822*.

Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. 2025. C3ot: Generating shorter chain-of-thought without compromising effectiveness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24312–24320.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. 2021. Deep reinforcement learning for autonomous driving: A survey. *IEEE transactions on intelligent transportation systems*, 23(6):4909–4926.

Derun Li, Changye Li, Yue Wang, Jianwei Ren, Xin Wen, Pengxiang Li, Leimeng Xu, Kun Zhan, Peng Jia, Xianpeng Lang, and 1 others. 2025a. Learning personalized driving styles via reinforcement learning from human feedback. *arXiv preprint arXiv:2503.10434*.

Yingyan Li, Yuqi Wang, Yang Liu, Jiawei He, Lue Fan, and Zhaoxiang Zhang. 2025b. End-to-end driving with online trajectory evaluation via bev world model. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 27137–27146. IEEE.

Yongkang Li, Kaixin Xiong, Xiangyu Guo, Fang Li, Sixu Yan, Gangwei Xu, Lijun Zhou, Long Chen, Haiyang Sun, Bing Wang, and 1 others. 2025c. Recogdrive: A reinforced cognitive framework for end-to-end autonomous driving. *arXiv preprint arXiv:2506.08052*.

Yun Li, Simon Thompson, Yidu Zhang, Ehsan Javanmardi, and Manabu Tsukada. 2026. An open-source modular benchmark for diffusion-based motion planning in closed-loop autonomous driving. *arXiv preprint arXiv:2603.01023*.

Zhenxin Li, Kailin Li, Shihao Wang, Shiyi Lan, Zhiding Yu, Yishen Ji, Zhiqi Li, Ziyue Zhu, Jan Kautz, Zuxuan Wu, and 1 others. 2024a. Hydra-mdp: End-to-end multimodal planning with multi-target hydra-distillation. *arXiv preprint arXiv:2406.06978*.

Zhenxin Li, Shihao Wang, Shiyi Lan, Zhiding Yu, Zuxuan Wu, and Jose M Alvarez. 2025d. Hydra-next: Robust closed-loop driving with open-loop training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 27305–27314.

Zhiqi Li, Zhiding Yu, Shiyi Lan, Jiahan Li, Jan Kautz, Tong Lu, and Jose M Alvarez. 2024b. Is ego status all you need for open-loop end-to-end autonomous driving? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14864–14873.

Bencheng Liao, Shaoyu Chen, Haoran Yin, Bo Jiang, Cheng Wang, Sixu Yan, Xinbang Zhang, Xiangyu Li, Ying Zhang, Qian Zhang, and 1 others. 2025. Diffusiondrive: Truncated diffusion model for end-to-end autonomous driving. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 12037–12047.

Hongbin Lin, Yiming Yang, Yifan Zhang, Chaoda Zheng, Jie Feng, Sheng Wang, Zhennan Wang, Shijia Chen, Boyang Wang, Yu Zhang, and 1 others. 2025. Futurex: Enhance end-to-end autonomous driving via latent chain-of-thought world model. *arXiv preprint arXiv:2512.11226*.

George Ling, Shanshan Zhong, and Richard Huang. 2026a. Agent skills: A data-driven analysis of claude skills for extending large language model functionality. *arXiv preprint arXiv:2602.08004*.

Guoming Ling, Zhongzhan Huang, Yupei Lin, Junxin Li, Shanshan Zhong, Hefeng Wu, and Liang Lin. 2026b. Neural chain-of-thought search: Searching the optimal reasoning path to enhance large language models. *arXiv preprint arXiv:2601.11340*.

Haochen Liu, Tianyu Li, Haohan Yang, Li Chen, Caojun Wang, Ke Guo, Haochen Tian, Hongchen Li, Hongyang Li, and Chen Lv. 2026a. Reinforced refinement with self-aware expansion for end-to-end autonomous driving. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Shu Liu, Wenlin Chen, Weihao Li, Zheng Wang, Lijin Yang, Jianing Huang, Yipin Zhang, Zhongzhan Huang, Ze Cheng, and Hao Yang. 2025. Bridgedrive: Diffusion bridge policy for closed-loop trajectory planning in autonomous driving. *arXiv preprint arXiv:2509.23589*.

Shuai Liu, Siheng Ren, Xiaoyao Zhu, Quanmin Liang, Zefeng Li, Qiang Li, Xin Hu, and Kai Huang. 2026b. Unidwm: Towards a unified driving world model via multifaceted representation learning. *arXiv preprint arXiv:2602.01536*.

Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Cheng Jiayang, Yue Zhang, Xipeng Qiu, and Zheng Zhang. 2024. Can language models learn to skip steps? *Advances in Neural Information Processing Systems*, 37:45359–45385.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025a. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*.

Yuechen Luo, Fang Li, Shaoqing Xu, Zhiyi Lai, Lei Yang, Qimao Chen, Ziang Luo, Zixun Xie, Shengyin Jiang, Jiaxin Liu, and 1 others. 2025b. Adathinkdrive: Adaptive thinking via reinforcement learning for autonomous driving. *arXiv preprint arXiv:2509.13769*.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025. Cot-valve: Length-compressible chain-of-thought tuning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6025–6035.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235.

Tergel Munkhbat, Namgyu Ho, Seo Hyun Kim, Yongjin Yang, Yujin Kim, and Se-Young Yun. 2025. Self-training elicits concise reasoning in large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 25127–25152.

Long Nguyen, Micha Fauth, Bernhard Jaeger, Daniel Dauner, Maximilian Igl, Andreas Geiger, and Kashyap Chitta. 2026. Lead: Minimizing learner-expert asymmetry in end-to-end driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 39775–39785.

Aditya Prakash, Kashyap Chitta, and Andreas Geiger. 2021. Multi-modal fusion transformer for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7077–7087.

Kangan Qian, Zhikun Ma, Yangfan He, Ziang Luo, Tianyu Shi, Tianze Zhu, Jiayin Li, Jianhui Wang, Ziyu Chen, Xiao He, and 1 others. 2024. Fasionad: Fast and slow fusion thinking systems for human-like autonomous driving with adaptive feedback. *arXiv preprint arXiv:2411.18013*.

Ziqing Qiao, Yongheng Deng, Jiali Zeng, Dong Wang, Lai Wei, Guanbo Wang, Fandong Meng, Jie Zhou, Ju Ren, and Yaoxue Zhang. 2025. Concise: Confidence-guided compression in step-by-step efficient reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 8021–8040.

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

Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. 2017. Deep reinforcement learning framework for autonomous driving. *arXiv preprint arXiv:1704.02532*.

Shuyao Shang, Bing Zhan, Yunfei Yan, Yuqi Wang, Yingyan Li, Yasong An, Xiaoman Wang, Jierui Liu, Lu Hou, Lue Fan, and 1 others. 2026. Dynvla: Learning world dynamics for action reasoning in autonomous driving. *arXiv preprint arXiv:2603.11041*.

Hao Shao, Yuxuan Hu, Letian Wang, Guanglu Song, Steven L Waslander, Yu Liu, and Hongsheng Li. 2024a. Lmdrive: Closed-loop end-to-end driving with large language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15120–15130.

Hao Shao, Letian Wang, Ruobing Chen, Hongsheng Li, and Yu Liu. 2023. Safety-enhanced autonomous driving using interpretable sensor fusion transformer. In *Conference on Robot Learning*, pages 726–737. PMLR.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024b. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, Zhaoxiang Liu, and Shiguo Lian. 2025. Dast: Difficulty-adaptive slow-thinking for large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 2322–2331.

Chonghao Sima, Katrin Renz, Kashyap Chitta, Li Chen, Hanxue Zhang, Chengen Xie, Jens Beißwenger, Ping Luo, Andreas Geiger, and Hongyang Li. 2024. Drivelm: Driving with graph visual question answering. In *European conference on computer vision*, pages 256–274. Springer.

Ziying Song, Caiyan Jia, Lin Liu, Hongyu Pan, Yongchang Zhang, Junming Wang, Xingyu Zhang, Shaoqing Xu, Lei Yang, and Yadan Luo. 2025. Don't shake the wheel: Momentum-aware planning in end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22432–22441.

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, and 1 others. 2020. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454.

Weizhe Tang, Junwei You, Jiaxi Liu, Zhaoyi Wang, Rui Gan, Zilin Huang, Feng Wei, and Bin Ran. 2026. Hermes: A holistic end-to-end risk-aware multimodal embodied system with vision-language models for long-tail autonomous driving. *arXiv preprint arXiv:2602.00993*.

Yingqi Tang, Zhuoran Xu, Zhaotie Meng, and Erkang Cheng. 2025. Hip-ad: Hierarchical and multi-granularity planning with deformable attention for autonomous driving in a single decoder. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 25605–25615.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Xiaoyu Tian, Junru Gu, Bailin Li, Yicheng Liu, Yang Wang, Zhiyong Zhao, Kun Zhan, Peng Jia, Xianpeng Lang, and Hang Zhao. 2024. Drivevlm: The convergence of autonomous driving and large vision-language models. *arXiv preprint arXiv:2402.12289*.

Sen Wang, Daoyuan Jia, and Xinshuo Weng. 2018. Deep reinforcement learning for autonomous driving. *arXiv preprint arXiv:1811.11329*.

Xinyang Wang, Qian Liu, Wenjie Ding, Zhao Yang, Wei Li, Chang Liu, Bailin Li, Kun Zhan, Xianpeng Lang, and Wei Chen. 2026. Unifying language-action understanding and generation for autonomous driving. *arXiv preprint arXiv:2603.01441*.

Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, and 1 others. 2023. Argoverse 2: Next generation datasets for self-driving perception and forecasting. *arXiv preprint arXiv:2301.00493*.

Penghao Wu, Xiaosong Jia, Li Chen, Junchi Yan, Hongyang Li, and Yu Qiao. 2022. Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. *Advances in Neural Information Processing Systems*, 35:6119–6132.

Yanhao Wu, Haoyang Zhang, Fei He, Rui Wu, Congpei Qiu, Liang Gao, Wei Ke, and Tong Zhang. 2026. Alignedrive: Aligned lateral-longitudinal planning for end-to-end autonomous driving. *arXiv preprint arXiv:2601.01762*.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. 2025. Tokenskip: Controllable chain-of-thought compression in llms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 3351–3363.

Jialong Xie, Yi Yang, Xing Jiabin, Jianyun Xu, Sheng Yang, and Fengyu Zhou. 2026. Deliberation meets reaction: A dual-expert VLA framework for autonomous driving.

Lijin Yang, Jianing Huang, Zhongzhan Huang, Shu Liu, and Hao Yang. 2026a. Judge, then drive: A critic-centric vision language action framework for autonomous driving. *arXiv preprint arXiv:2604.27366*.

Zhenjie Yang, Xiaosong Jia, Qifeng Li, Xue Yang, Maoqing Yao, and Junchi Yan. 2026b. Raw2drive: Reinforcement learning with aligned world models for end-to-end autonomous driving (in carla v2). *Advances in Neural Information Processing Systems*, 38:134122–134147.

Rajeev Yasarla, Deepti Hegde, Shizhong Han, Hsin-Pai Cheng, Yunxiao Shi, Meysam Sadeghigooghari, Shweta Mahajan, Apratim Bhattacharyya, Litian Liu, Risheek Garrepalli, and 1 others. 2026. Generative scenario rollouts for end-to-end autonomous driving. *arXiv preprint arXiv:2601.11475*.

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. 2025. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*.

Zihan You, Hongwei Liu, Chenxu Dang, Zhe Wang, Sining Ang, Aoqi Wang, and Yan Wang. 2026. Samoe-vla: A scene adaptive mixture-of-experts vision-language-action model for autonomous driving. *arXiv preprint arXiv:2603.08113*.

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. 2024. Distilling system 2 into system 1. *arXiv preprint arXiv:2407.06023*.

Dapeng Zhang, Zhenlong Yuan, Zhangquan Chen, Chih-Ting Liao, Yinda Chen, Fei Shen, Qingguo Zhou, and Tat-Seng Chua. 2025a. Reasoning-vla: A fast and general vision-language-action reasoning model for autonomous driving. *arXiv preprint arXiv:2511.19912*.

Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. 2025b. Adaptthink: Reasoning models can learn when to think. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 3716–3730.

Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. 2024. Vision-language models for vision tasks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5625–5644.

Wenzhao Zheng, Ruiqi Song, Xianda Guo, Chenming Zhang, and Long Chen. 2024. Genad: Generative end-to-end autonomous driving. In *European Conference on Computer Vision*, pages 87–104. Springer.

Shanshan Zhong, Zhongzhan Huang, Shanghua Gao, Wushao Wen, Liang Lin, Marinka Zitnik, and Pan Zhou. 2024. Let's think outside the box: Exploring leap-of-thought in large language models with creative humor generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13246–13257.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. 2022. Learning to prompt for vision-language models. *International journal of computer vision*, 130(9):2337–2348.

Zewei Zhou, Tianhui Cai, Seth Zhao, Yun Zhang, Zhiyu Huang, Bolei Zhou, and Jiaqi Ma. 2026. Autovla: A vision-language-action model for end-to-end autonomous driving with adaptive reasoning and reinforcement fine-tuning. *Advances in Neural Information Processing Systems*, 38:27920–27956.

Julian Zimmerlin, Jens Beißwenger, Bernhard Jaeger, Andreas Geiger, and Kashyap Chitta. 2024. Hidden biases of end-to-end driving datasets. *arXiv preprint arXiv:2412.09602*.

Contents

A The Novelty and Contribution of BLUE 15
A.1 Novelty of BLUE 15
B Related Work 16
B.1 End-to-End Autonomous Driving . . . 16
B.2 Adaptive Reasoning in Driving VLA . 16
B.3 Efficient Reasoning in LLMs 17
B.3.1 RL with Length Reward Design . 17
B.3.2 SFT with Variable-Length CoT . . 17
B.4 Autonomous Driving Benchmarks . . 17
B.5 Comparison with Related Methods . . 18
B.5.1 Adaptive Reasoning in VLA . . . 18
B.5.2 Efficient Reasoning in LLM . . . 19
C Additional Experimental Details 19
C.1 Details of Data Splits 19
C.2 Details of Benchmarks Considered . . 20
C.2.1 Details of Bench2Drive 20
C.2.2 Details of Longest6 v2 20
C.2.3 Details of Fail2Drive 21
C.2.4 Details of NAVSIM 21
C.3 Details of Baselines Considered . . . 21
C.4 Details of Label Construction 23
C.4.1 Route-Level Labels 23
C.4.2 Frame-Level Labels 23
C.4.3 Temporal Redundancy Cleaning . 24
C.5 Implementation and Hyperparameters . 24
C.5.1 Data Collection 24
C.5.2 Gate Architecture and Training . . 24
C.5.3 Computational Cost 24
C.6 Details of Baseline Gates 24
D Additional Experimental Results . . . 25
D.1 Full Bench2Drive Comparison 25
D.2 Language Impact Analysis 25
D.2.1 Experimental Settings 25
D.2.2 Statistical Methods 26
D.2.3 Detailed Results 27
D.2.4 Discussion 27
E Additional Analysis 28
E.1 Language Content Visualization . . . 28
F Additional Statements 28

A The Novelty and Contribution of BLUE

In this section, we summarize the main novelty of BLUE and explain how does this work contribute to the NLP community. Detailed comparisons with related methods are provided in Section B.5.

A.1 Novelty of BLUE

BLUE is related to recent work on language-augmented driving, efficient reasoning, and adaptive computation. We highlight five aspects that distinguish BLUE, while deferring detailed method-by-method comparisons to Section B.5.

Systematic diagnosis of language utility. Prior work on adaptive reasoning focuses on improving, accelerating, or selectively invoking language generation, but none has systematically quantified how generated language affects closed-loop driving outcomes. We conduct large-scale closed-loop evaluations on hundreds of routes and statistically categorize each route as language-helpful, language-neutral, or language-harmful. The results show that language improves driving on only a minority of routes, actively degrades it on another subset, and has no clear effect on the remaining majority. This finding offers a new perspective for the community: selective language use in VLA driving should not only reduce unnecessary computation but also actively prevent harmful language generation.

Hidden-state language-utility signal. Existing methods rely on reinforcement learning rewards, scene complexity features, or architectural modifications to learn when to reason, all of which require retraining or modifying the VLA backbone. BLUE discovers that the pretrained hidden states of a frozen VLA potentially already encode whether language generation will benefit driving at the current frame. This means the gating decision does not require redesigning the model or additional training of the backbone. The gate simply reads out a signal that already exists inside the pretrained representations, keeping the original VLA unchanged and avoiding the cost of backbone-level retraining.

Low-cost label collection. Existing adaptive-reasoning methods typically require constructing reward functions or curating specialized training data. BLUE derives its training labels by simply comparing route success when the VLA generates language versus when it predicts actions directly. This process requires no human annotation and no

reward engineering. In autonomous driving, deploying or validating a model already involves collecting driving outcomes under different configurations. BLUE reuses these routine evaluation results as supervision, making label collection a natural byproduct of the standard development pipeline rather than an additional burden.

State-of-the-art results with full reproducibility. Despite training only a 0.11M-parameter MLP gate on a frozen VLA backbone, BLUE achieves state-of-the-art closed-loop driving results on both Bench2Drive and Longest6 v2 while delivering 2.54× inference speedup. Many concurrent driving methods remain closed-source, making their results difficult to reproduce or build upon. We fully release the code, training data, model checkpoints, and evaluation logs, providing the community with a fully reproducible baseline for future research on better language use in VLA driving.

Toward better language use in vision and robotics. As language models are increasingly used in visual and robotic systems, language generation must improve downstream behavior under latency constraints. Existing approaches mainly optimize the amount or speed of generation. BLUE demonstrates a complementary principle: better language use requires deciding when to generate, not only how to generate. In embodied settings, unnecessary language can slow the system and steer actions in harmful ways. Maximizing language's net positive impact therefore requires generation decisions that adapt to each input. This insight extends to any deployed system where language serves as an intermediate computation.

B Related Work

BLUE sits at the intersection of end-to-end autonomous driving, adaptive reasoning, and driving evaluation. We first review end-to-end driving methods with a focus on language-augmented models (§B.1), then discuss efficient reasoning techniques in both VLA driving (§B.2) and LLMs (§B.3), followed by the benchmarks used for evaluation (§B.4). Finally, we provide a detailed comparison between BLUE and existing methods (§B.5).

B.1 End-to-End Autonomous Driving

Autonomous driving has evolved from modular pipelines that chain perception, prediction, and planning into end-to-end models that directly map sensor inputs to driving actions (Chen et al., 2024a; Hu et al., 2023b; Wu et al., 2022; Li et al., 2025b). Within this paradigm, diverse architectural designs have emerged, including multi-modal transformer fusion (Prakash et al., 2021; Chitta et al., 2022; Shao et al., 2023), vectorized scene representation for planning (Hu et al., 2023b; Jiang et al., 2023; Li et al., 2024b; Jia et al., 2025), learning-based planner supervision (Dauner et al., 2023; Li et al., 2024a; Wu et al., 2026; Li et al., 2025a), and diffusion-based or generative trajectory prediction (Liao et al., 2025; Zheng et al., 2024; Liu et al., 2025). A growing line of work further integrates natural language into the driving loop through vision-language models and vision-language-action models (Zhang et al., 2024; Zhou et al., 2022; Zhong et al., 2024; Huang et al., 2025b; Wang et al., 2026), which generate scene descriptions or reasoning chains as intermediate representations before producing control outputs (Sima et al., 2024; Tian et al., 2024; Shao et al., 2024a; Hwang et al., 2024; Jiang et al., 2024; Tang et al., 2026; Ling et al., 2026a). Complementary directions include world models for predictive simulation (Hu et al., 2023a; Gao et al., 2024; Liu et al., 2026b) and reinforcement learning for driving policy optimization (Kiran et al., 2021; Sallab et al., 2017; Wang et al., 2018).

B.2 Adaptive Reasoning in Driving VLA

Recent VLA driving models typically generate language reasoning before predicting actions, but language generation introduces significant inference overhead. Several concurrent methods have explored strategies to reduce this cost. One line of work trains VLA models to select reasoning depth adaptively. AutoVLA (Zhou et al., 2026) builds a unified autoregressive VLA with physical action tokenization and applies supervised fine-tuning followed by GRPO-based reinforcement fine-tuning to reduce reasoning in straightforward scenarios. AdaThinkDrive (Luo et al., 2025b) trains a dual-mode think and non-think policy through supervised learning and GRPO with an adaptive think reward. Another line of work introduces multi-system architectures. DE-Driver (Xie et al., 2026) designs a dual-expert VLA with a scene-aware router that dispatches inputs to a reactive or deliberative expert. SAMoE-VLA (You et al., 2026) extends this idea with scene-adaptive mixture-of-experts routing, and FASIONAD (Qian et al., 2024) adopts a dual-system framework where a fast sys-

tem handles routine navigation while a slow system reasons from visual prompts and provides feedback in challenging situations. A related direction seeks alternative reasoning representations. FutureX (Lin et al., 2025) uses an auto-think switch to choose between instant planning and latent world-model reasoning, invoking CoT-guided latent rollout only when additional reasoning is needed. DynVLA (Shang et al., 2026) introduces a dynamics chain-of-thought that compresses future world evolution into compact dynamics tokens. From the efficiency perspective, FastDriveCoT (Gu et al., 2026) applies parallel decoding to structured chain-of-thought templates, and Reasoning-VLA (Zhang et al., 2025a) replaces autoregressive action decoding with learnable action queries for parallel trajectory generation. We provide a detailed comparison between BLUE and these methods in §B.5.1.

B.3 Efficient Reasoning in LLMs

BLUE detects when language generation is needed in VLA driving. This question is related to efficient reasoning in NLP, where recent methods control the length of chain-of-thought through RL with length rewards or SFT on variable-length reasoning data.

B.3.1 RL with Length Reward Design

Early RL training for reasoning models focuses on accuracy rewards, which often leads to unnecessarily verbose chains of thought. To address this, recent methods incorporate length-based penalties into the reward function so that shorter correct answers receive higher scores. Kimi k1.5 (Team et al., 2025) adds a length penalty to its policy optimization to control long reasoning activations. O1-Pruner (Luo et al., 2025a) introduces a length-harmonizing reward with PPO, optimizing the ratio of reasoning lengths between a reference model and the student. L1 (Aggarwal and Welleck, 2025) appends explicit token-budget constraints to the input before applying GRPO (Shao et al., 2024b). Demystifying Long CoT (Yeo et al., 2025) proposes a cosine-shaped reward that penalizes excessive length while stabilizing training. DAST (Shen et al., 2025) constructs length-preference data and trains with SimPO (Meng et al., 2024) to adapt reasoning depth to problem difficulty. Arora et al. (Arora and Zanette, 2026) condition length rewards on correctness, assigning higher scores to shorter correct solutions. Other representative efforts include AdaptThink (Zhang et al., 2025b), S-GRPO (Dai et al., 2026), and ConciseRL (Dumitru et al., 2025). These methods share the goal of producing concise reasoning in text-based question answering and mathematics. BLUE addresses a related but distinct setting: instead of shortening textual reasoning, it decides whether to activate language generation at all in an embodied VLA driving model.

B.3.2 SFT with Variable-Length CoT

Beyond RL, supervised fine-tuning on curated variable-length reasoning data provides another route to efficient reasoning. These methods differ mainly in how the short chains are constructed. In the post-reasoning approach, Distilling System 2 into System 1 (Yu et al., 2024) removes the reasoning process entirely and distills only the final answer. C3oT (Kang et al., 2025) uses GPT-4 as a compressor to shorten reasoning while retaining key information. TokenSkip (Xia et al., 2025) estimates the semantic importance of each reasoning segment and removes low-importance tokens. In the during-reasoning approach, Learn to Skip (Liu et al., 2024) first creates concise solutions by manually merging or removing steps, then trains the model to intrinsically skip steps during inference. Token-Budget (Han et al., 2025) uses binary search to find the optimal token budget and trains the model to follow it. Self-Training (Munkhbat et al., 2025) samples multiple reasoning paths and selects the shortest correct one as training data. CoT-Valve (Ma et al., 2025) progressively mixes parameters of long-reasoning and non-reasoning models to generate variable-length training data. Other related works include ReCUT (Jin et al., 2025), ConCISE (Qiao et al., 2025) and NCoTS (Ling et al., 2026b). Like RL-based methods, these approaches focus on compressing textual reasoning chains in language tasks. BLUE operates at a coarser granularity: rather than shortening the generated language, it uses a lightweight gate to decide whether language generation should be activated for a given frame.

B.4 Autonomous Driving Benchmarks

Evaluating autonomous driving models falls into two broad categories depending on whether the model's own outputs influence future states. Open-loop benchmarks, including nuScenes (Caesar et al., 2020), Waymo Open (Sun et al., 2020), and Argoverse 2 (Wilson et al., 2023), measure prediction quality against recorded ground truth but do not let the model's actions affect subse-

Method	What is Adapted	Core Mechanism	Frozen	Supervision
AutoVLA (Zhou et al., 2026)	Reasoning depth	Action token. + SFT + GRPO	×	RL reward
AdaThinkDrive (Luo et al., 2025b)	Think / non-think	Dual-mode SFT + GRPO	×	Adaptive think reward
DE-Driver (Xie et al., 2026)	Reactive / delib. expert	Dual-expert + scene router	×	Scene-aware routing
SAMoE-VLA (You et al., 2026)	Expert assignment	Scene-adaptive MoE	×	Scene features
FASIONAD (Qian et al., 2024)	Fast / slow system	Dual-system + VLM feedback	×	Confidence score
FutureX (Lin et al., 2025)	Instant / latent thinking	Auto-think + world model	×	World-model rollout
DynVLA (Shang et al., 2026)	Text / dynamics CoT	Dynamics token prediction	×	SFT on dyn. tokens
FastDriveCoT (Gu et al., 2026)	CoT decoding speed	Parallel structured decoding	×	Template structure
Reasoning-VLA (Zhang et al., 2025a)	Action decoding	Action queries + parallel gen.	×	SFT
BLUE (ours)	Language gen. (on / off)	Hidden-state gate (0.11M)	✓	Language utility

Table 12: Comparison of BLUE with related adaptive reasoning methods for VLA driving. **Frozen** indicates whether the original VLA backbone weights remain frozen. BLUE is the only method that performs post-hoc language gating on a frozen VLA, with labels derived from closed-loop driving outcomes.

quent observations. As a result, prediction errors that would compound during actual driving remain hidden in open-loop evaluation. Closed-loop benchmarks fill this gap by requiring the model to drive full routes inside a simulator or on replayed logs, where cumulative effects on driving outcomes such as route completion and collision avoidance can be directly measured. On the simulator side, CARLA (Dosovitskiy et al., 2017) provides a configurable urban environment that supports diverse traffic and weather conditions. Built on CARLA, Bench2Drive (Jia et al., 2024) defines 220 routes across 44 scenario categories and evaluates models on multiple metrics such as success rate and driving score, covering five distinct driving skills. Longest6 v2 (autonomousvision, 2026) contains 36 long routes of 1–2 km each and evaluates sustained driving quality over extended distances. Fail2Drive (Gerstenecker et al., 2026) evaluates generalization across 100 standard routes and 100 rare, challenging long-tail routes, such as those involving animal crossings or fully blocked roads. On the log-replay side, nuPlan (Caesar et al., 2021), WayMax (Gulino et al., 2023), NavSim (Dauner et al., 2024), and OpenAD (Li et al., 2026) replay real-world driving logs with reactive or non-reactive traffic agents. We evaluate BLUE on three closed-loop benchmarks: Bench2Drive, Longest6 v2, and Fail2Drive. Together, they provide complementary evaluations of multi-scenario driving, long-horizon driving, and generalization to challenging long-tail scenarios. To further assess BLUE on real-world driving data, we also conduct experiments on NAVSIM (Dauner et al., 2024). Experiments across all four benchmarks broadly validate the effectiveness of BLUE.

B.5 Comparison with Related Methods

We now compare BLUE with adaptive reasoning methods in VLA driving (§B.5.1) and with efficient reasoning methods in LLMs (§B.5.2).

B.5.1 Adaptive Reasoning in VLA

Section B.2 surveys the methods covered here. Table 12 provides a structured comparison of BLUE with these methods. As shown in Table 12, BLUE differs from existing methods in four aspects.

First, existing adaptive reasoning methods focus on learning when to activate longer or shorter reasoning, but none of them systematically examines how generated language affects closed-loop driving performance. BLUE fills this gap by conducting extensive evaluations and quantitatively categorizing language impact into helpful, neutral, and harmful cases, offering insights into when and why language generation should be selectively applied.

Second, all prior methods require modifying the VLA backbone or introducing new architectural components, whereas BLUE keeps the VLA entirely frozen and trains only a 0.11M-parameter gate. The minimal parameter count means the gate can be trained in minutes on a single GPU, adds negligible inference overhead, and avoids overfitting on limited calibration data, making BLUE a lightweight plug-in applicable to many existing VLA model without retraining.

Third, existing methods rely on reinforcement learning with custom reward functions, scene-level features, or new training objectives, all of which demand additional supervision and pipeline changes.

BLUE instead discovers that pretrained VLA hidden states potentially already encode whether language generation will benefit driving, and the gate simply reads out this existing signal. The training labels are collected automatically by running the VLA and comparing driving outcomes, requiring no human annotation, no reward engineering, and minimal additional cost, as the data collection can be integrated into the routine road testing that deployed driving models already undergo.

Fourth, existing adaptive VLA methods such as AutoVLA and AdaThinkDrive (Zhou et al., 2026; Luo et al., 2025b) base their reasoning decisions on scenario complexity, activating longer reasoning in difficult scenes and reducing it in simple ones. However, as demonstrated in Section 5, complexity-based gates perform comparably to simple kinematic heuristics and remain far below BLUE. The underlying reason, analyzed in detail in Appendix D.2, is that whether language helps depends not only on the scenario but also on how a specific model processes and utilizes language. The same scenario can be language-helpful under one model yet language-harmful under another, making complexity an unreliable proxy. BLUE addresses this limitation by conditioning the gating decision on model-specific hidden states, which jointly encode perceptual context and the model's internal readiness to benefit from language generation.

B.5.2 Efficient Reasoning in LLM

Section B.3 surveys RL-based and SFT-based methods for controlling reasoning length in LLMs. BLUE shares the motivation of avoiding unnecessary reasoning but differs in three ways. First, LLM methods (Team et al., 2025; Luo et al., 2025a; Huang et al., 2025a; Yeo et al., 2025) control how long a reasoning chain should be, producing shorter or longer traces depending on problem difficulty. BLUE makes a binary decision: whether language generation should be activated at all for a given driving frame. This coarser formulation suits VLA driving, where the two inference modes produce qualitatively different action distributions rather than merely longer or shorter textual outputs. Second, in text-based tasks the cost of reasoning is primarily token count and latency, whereas in closed-loop driving, generated language is an intermediate computation that alters the action trajectory. Unnecessary language can therefore actively degrade driving performance, a harmful effect that BLUE systematically quantifies and that has no direct counterpart in text-based settings. Third, LLM methods modify the language model itself through RL or SFT, while BLUE keeps the VLA backbone frozen and trains only a 0.11M-parameter gate supervised by paired closed-loop driving outcomes rather than task accuracy or token cost.

C Additional Experimental Details

This section provides implementation and evaluation details needed for reproducibility.

C.1 Details of Data Splits

We follow the standard data splits and ensure full separation between training and evaluation.

Training Set. For all CARLA simulation experiments, data collection for BLUE, including hidden-state extraction, label construction, and gate training, is performed exclusively on routes from the SimLingo training set. For each training route, we run language mode and direct action mode separately through repeated experiments, recording the per-route success rate of each mode. The success rate gap between the two modes determines the training label for that route. We also extract the hidden state at every frame during these runs, which serves as the input feature for gate training. Fail2Drive is used only for direct evaluation and not for training. For NAVSIM, we train on `navtrain`. The specific route counts, seed configurations, and sampling strategy are detailed in Section C.5.

Evaluation Set. We evaluate BLUE on the standard Bench2Drive test split, the full Longest6 v2 benchmark, and Fail2Drive. These CARLA evaluation routes are entirely disjoint from the training routes, and our split is consistent with prior work to ensure fair comparison. No training data, labels, or hidden states are derived from these evaluation routes. For NAVSIM, we evaluate on `navtest`.

Hyperparameter Selection. We do not perform systematic hyperparameter search. All hyperparameters are set with simple heuristics. For example, gate threshold θ=0.66 follows from the observation that language impact falls into three natural categories: language-harmful, language-neutral, and language-helpful. These three categories map to three equal intervals on the gate output range $[0, 1]$, so we place the threshold at the boundary of the upper interval. This simple choice already yields strong performance. We provide threshold sensitivity analysis in Section 5.

C.2 Details of Benchmarks Considered

We evaluate BLUE on three complementary closed-loop benchmarks built on the CARLA simulator (Dosovitskiy et al., 2017). Bench2Drive tests multi-scenario driving over scenario-focused routes, Longest6 v2 evaluates sustained driving quality over longer routes, and Fail2Drive (Gerstenecker et al., 2026) assesses generalization across 100 standard routes and 100 rare, challenging long-tail routes. Together, they cover scenario diversity, route length, and long-tail generalization. We further evaluate BLUE on NAVSIM (Dauner et al., 2024), which uses real-world driving data, to assess its effectiveness beyond simulation. We provide further details on all four benchmarks below.

C.2.1 Details of Bench2Drive

Bench2Drive (Jia et al., 2024) is a closed-loop benchmark built on CARLA v2 for evaluating end-to-end autonomous driving systems across multiple driving skills. It defines 220 evaluation routes distributed across 44 interactive scenario categories, 23 weather conditions, and 12 towns. Each route contains a single safety-critical scenario such as cut-in, overtaking, construction obstacle, or pedestrian crossing. The ego vehicle receives raw sensor inputs and target waypoints, and must drive from the source to the destination within an allotted time without traffic violations.

Multi-Ability Evaluation. Bench2Drive groups the 44 scenarios into five high-level driving skills: Merging, Overtaking, Emergency Brake, Give Way, and Traffic Sign. Each skill score is the success rate over the corresponding subset of routes, and their average forms the multi-ability mean. This design enables disentangled assessment of individual driving capabilities.

Metrics. Bench2Drive reports four evaluation metrics. Success Rate and Driving Score capture goal-achieving ability, while Driving Efficiency and Driving Smoothness measure driving quality.

Success Rate (SR). Success Rate measures the proportion of routes completed within the allotted time without any traffic violation. A route is successful only if the ego vehicle reaches its destination with no recorded infraction. Formally, $\text{SR} = n_{\text{success}}/n_{\text{total}}$, where n_{success} and n_{total} denote the number of successful and total routes.

Driving Score (DS). Driving Score follows the CARLA Leaderboard protocol and combines route completion with infraction penalties:

$$\text{DS} = \frac{1}{n_{\text{total}}} \sum\nolimits_{i=1}^{n_{\text{total}}} \text{RC}_i \cdot \prod\nolimits_j p_{i,j}, \qquad (3)$$

where $\text{RC}_i \in [0, 100]$ is the route completion percentage for route i and $p_{i,j} \in (0, 1]$ is the penalty factor for the j-th infraction. Penalties compound multiplicatively, so a single serious infraction can substantially reduce the score.

Driving Efficiency. Driving Efficiency evaluates whether the ego vehicle maintains a reasonable speed relative to surrounding traffic. At each checkpoint, the ego speed is compared to the average speed of nearby vehicles, yielding a speed percentage. Bench2Drive checks speed every 5% of the route length across 20 checkpoints to reduce measurement variance. The final metric is the mean speed percentage across all checkpoints: $\overline{v}_{\%} = (\sum_i v_{\%,i})/C$, where C is the total number of checkpoints.

Driving Smoothness. Driving Smoothness evaluates trajectory comfort following the nuPlan protocol (Caesar et al., 2021). At each frame, six kinematic variables are checked against expert-derived thresholds: longitudinal acceleration, lateral acceleration, yaw rate, yaw acceleration, longitudinal jerk, and jerk magnitude. A frame is smooth only if all six variables fall within their bounds. To mitigate the effect of necessary reactions such as emergency braking, Bench2Drive segments the trajectory into intervals of 20 timesteps and evaluates smoothness per segment. The final score is the ratio of smooth segments to total segments: $\text{Smoothness} = S_{\text{smooth}}/S_{\text{total}}$.

C.2.2 Details of Longest6 v2

Longest6 v2 (autonomousvision, 2026) is a closed-loop benchmark derived from the original Longest6 benchmark (Chitta et al., 2022). It consists of 36 long routes of 1–2 km each and 7 scenario types. It evaluates whether a driving model can sustain safe and effective performance over longer driving horizons, where early errors and infractions can propagate and compound. Longest6 v2 reports three standard CARLA Leaderboard metrics.

Driving Score (DS). Driving Score is defined identically to the Bench2Drive formulation: the average over all routes of the route completion percentage multiplied by the cumulative infraction penalty. Over longer routes, the multiplicative penalty structure makes Driving Score more

sensitive to infractions, since more events can accumulate along the route.

Route Completion (RC). Route Completion measures the average percentage of the prescribed route distance that the ego vehicle successfully traverses before termination. It captures how far the model can drive regardless of infractions.

Infraction Score (IS). Infraction Score isolates the penalty component by reporting the average product of all per-route infraction multipliers: $\text{IS} = \frac{1}{n_{\text{total}}} \sum_i \prod_j p_{i,j}$. A higher Infraction Score indicates fewer and less severe violations. Together with Route Completion, Infraction Score enables diagnosis of whether low Driving Score stems from incomplete routes or from frequent infractions.

C.2.3 Details of Fail2Drive

Fail2Drive (Gerstenecker et al., 2026) is a closed-loop benchmark built on CARLA 0.9.15 for evaluating generalization under controlled distribution shifts. It contains 200 short routes in Town 13, with an average length of 219 meters. The routes form 100 pairs, each consisting of an in-distribution route with a standard CARLA scenario and a generalization route with a targeted long-tail shift. Within each pair, the road geometry, spawn points, traffic conditions, and driving objective remain fixed, allowing the performance difference to isolate the effect of the distribution shift.

Generalization Scenarios. Fail2Drive introduces 17 scenario classes organized into four categories: robustness, visual generalization for lateral control, visual generalization for longitudinal control, and behavioral generalization. These categories test whether driving models remain reliable under irrelevant visual changes, unseen obstacle appearances and layouts, novel causal objects such as crossing animals, and unfamiliar behaviors such as stopping for a fully blocked road.

Metrics. Fail2Drive follows the Bench2Drive protocol and reports Driving Score and Success Rate on both the in-distribution and generalization routes. Because its routes are short and usually finishable, RC is not reported as a separate metric. Fail2Drive additionally summarizes the two primary metrics using their harmonic mean:

$$\text{HM} = \frac{2 \cdot \text{DS} \cdot \text{SR}}{\text{DS} + \text{SR}}. \quad (4)$$

For each metric M, the relative generalization change is computed as $100(M_{\text{gen}} - M_{\text{ID}})/M_{\text{ID}}$. This paired comparison directly measures how much performance changes when the model encounters a controlled long-tail shift.

C.2.4 Details of NAVSIM

NAVSIM (Dauner et al., 2024) is a data-driven, non-reactive benchmark for evaluating autonomous driving planners on real-world sensor data. It is built on OpenScene, a redistribution of the nuPlan dataset (Caesar et al., 2021), and provides standardized `navtrain` and `navtest` splits with approximately 103K and 12K samples, respectively. Each sample may include multi-view camera images, LiDAR, ego motion states, and a navigation goal.

Non-Reactive Simulation. Given the initial sensor observation, an agent predicts a trajectory over a 4-second horizon and is not queried again during the rollout. An LQR controller and a kinematic bicycle model execute the predicted trajectory at 10 Hz, while surrounding agents follow their recorded trajectories. This design enables scalable and deterministic evaluation on real-world data, while simulation-based outcomes provide a closer approximation to closed-loop driving quality than displacement errors alone.

Predictive Driver Model Score (PDMS). NAVSIM evaluates safety, progress, and comfort through five normalized subscores: No At-Fault Collision (NC), Drivable Area Compliance (DAC), Time to Collision (TTC), Ego Progress (EP), and Comfort (C). NC and DAC serve as multiplicative penalties, while TTC, EP, and C are combined using a weighted average:

$$\text{PDMS} = \text{NC} \cdot \text{DAC} \cdot \frac{5\text{EP} + 5\text{TTC} + 2\text{C}}{12}. \quad (5)$$

NC penalizes at-fault collisions, DAC measures whether the trajectory remains within the drivable area, TTC captures near-term collision risk, EP measures progress relative to a safe reference planner, and C evaluates acceleration and jerk. We also report Driving Direction Compliance as an auxiliary diagnostic metric. NAVSIM complements the three CARLA benchmarks by testing BLUE on recorded real-world observations, although its non-reactive evaluation does not model long-term interaction or compounding closed-loop errors.

C.3 Details of Baselines Considered

We compare BLUE against a wide range of published methods spanning end-to-end driving and

vision-language-action approaches. Table 1 lists the sensor configuration and training labels for each method. As shown in the table, most baselines rely on surround cameras, LiDAR, or dense auxiliary labels such as 3D object detection, HD maps, semantic segmentation, and depth. In contrast, BLUE uses only front-view camera without LiDAR and requires only language annotations, yet achieves the best closed-loop results on two benchmarks. Below we briefly describe each baseline.

UniAD-Base (Hu et al., 2023b) unifies perception, prediction, and planning into a single network, where all tasks communicate through unified query interfaces and are jointly optimized toward planning. It uses six surround cameras and trains with 3D object detection, map, and segmentation labels.

TF++ (Zimmerlin et al., 2024) analyzes training data biases and proposes a label-change-based frame selection criterion to compress datasets without losing important information. It uses a front-view camera with LiDAR and trains with object detection, map, segmentation, and depth labels.

MomAD (Song et al., 2025) introduces trajectory momentum and perception momentum to stabilize long-horizon planning by selecting planning queries topologically consistent with historical paths and fusing them with historical context. It uses six cameras with object detection and map labels.

DriveTransformer (Jia et al., 2025) replaces the sequential perception-prediction-planning pipeline with parallel task interaction, where agent, map, and planning queries directly attend to each other at every block. It uses six cameras with object detection and map labels.

Hydra-NeXt (Li et al., 2025d) adopts a multi-branch framework that unifies trajectory prediction, control prediction, and trajectory refinement to bridge the gap between open-loop training and closed-loop driving. It uses two cameras without auxiliary perception labels.

Raw2Drive (Yang et al., 2026b) follows a dual-stream model-based reinforcement learning approach, first training a privileged world model and then aligning a raw-sensor world model to it through a guidance mechanism. It uses six cameras with map and segmentation labels.

DiffusionDrive (Liao et al., 2025) applies a truncated diffusion policy with multi-mode anchors, compressing the denoising process to produce diverse driving actions in only a few steps. It uses multiple cameras with LiDAR and trains with object and segmentation labels.

ORION (Fu et al., 2025) bridges semantic reasoning and numerical trajectory output by combining a QT-Former for long-term context aggregation, a large language model for scene reasoning, and a generative planner for trajectory prediction. It uses six cameras with object detection, map, and language labels.

AutoVLA (Zhou et al., 2026) unifies reasoning and action generation within a single autoregressive model by tokenizing continuous trajectories into discrete action tokens. It supports fast and slow thinking modes and uses GRPO-based reinforcement fine-tuning to reduce unnecessary reasoning. It uses a front-view camera with language labels.

SimLingo (Renz et al., 2025) is a camera-only vision-language model that jointly handles closed-loop driving, vision-language understanding, and language-action alignment. It uses a single front-view camera with language annotations and serves as the primary backbone for BLUE.

HiP-AD (Tang et al., 2025) introduces multi-granularity planning queries that integrate spatial, temporal, and driving-style waypoints, and uses deformable attention to retrieve image features based on physical trajectory locations. It uses six cameras with object detection, map, and depth labels.

ReCogDrive (Li et al., 2025c) combines an autoregressive cognitive model with a diffusion planner, where the former provides driving reasoning priors and the latter generates continuous trajectories. It uses six cameras with language labels.

GeRo (Yasarla et al., 2026) extends VLA models with language-conditioned autoregressive generation of future traffic scenes, encoding ego and agent dynamics into shared latent tokens and stabilizing long-horizon rollouts with a consistency loss. It uses six cameras with object detection, map, and language labels.

DeLL (Du et al., 2026) addresses lifelong learning in end-to-end driving through a Dirichlet process mixture model that builds dynamic knowledge

spaces, combined with front-door causal adjustment to suppress spurious correlations. It uses a front-view camera with LiDAR and trains with object detection and segmentation labels.

R2SE (Liu et al., 2026a) proposes a three-stage pipeline: generalist pretraining with hard-case identification, residual reinforcement fine-tuning on difficult scenarios, and self-aware adapter expansion that routes between generalist and specialist policies at test time. It uses a front-view camera with LiDAR and trains with object detection, map, segmentation, and depth labels.

AutoMoT (Huang et al., 2026) unifies reasoning and action generation within a mixture-of-transformers architecture, where a frozen understanding expert and a high-frequency action expert share a joint attention space for asynchronous fast-slow inference. It uses a front-view camera with LiDAR.

BevAD (Holtz et al., 2026) re-examines common architectural patterns for closed-loop driving and identifies effective combinations of spatial bottleneck compression, decoupled trajectory representation, and diffusion-based planning. It uses six cameras with object detection labels.

CriticVLA (Yang et al., 2026a) extends VLA models from acting to judging: it generates a rough trajectory and then uses the VLA as a critic to evaluate and refine the plan through single-step optimization. It uses a single front-view camera with language labels and serves as a secondary backbone for validating BLUE.

TakeVLA (Gao et al., 2026) improves VLA driving through post-training on expert takeover data, shifting language supervision to the period before takeover moments so that the model learns to anticipate hazards early. It uses a single front-view camera with language labels.

C.4 Details of Label Construction

The gate requires binary labels indicating whether each frame benefits from language generation. A key advantage of our labeling approach is that it requires no human annotation: all labels are derived automatically from closed-loop evaluation outcomes by comparing the two modes. We construct labels at two granularities, route-level and frame-level, and apply temporal redundancy cleaning before training.

C.4.1 Route-Level Labels

We run language mode and direct action mode on each training route with $|\mathcal{S}|$=5 random seeds. The cross-seed success rate for mode m on route r is $\overline{\mathrm{SR}}_m^{(r)} = \frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}} \mathrm{SR}_m^{(r,s)}$. As defined in Eq. 1, a route receives label y_r=1 when the language advantage exceeds a margin threshold:

$$\Delta\overline{\mathrm{SR}}_r = \overline{\mathrm{SR}}_{\text{lang}}^{(r)} - \overline{\mathrm{SR}}_{\text{direct}}^{(r)} > \tau, \tag{6}$$

where τ=10% ensures that language mode is activated only when it provides a sufficiently large performance gain. Routes that do not meet this condition are labeled y_r=0, defaulting to the faster direct action mode. Under route-level labeling, all frames within a route share the same label y_r.

C.4.2 Frame-Level Labels

Route-level labels assign a uniform label to all frames in a route, but in practice, even on a language-beneficial route, only certain segments truly require language. To provide finer supervision, we identify critical regions $\mathcal{C}_r$ where the two modes exhibit the largest behavioral divergence. The core idea is straightforward: we spatially compare how the vehicle behaves under the two modes and mark the locations where their behaviors differ the most. Concretely, for each language-beneficial route, we collect 2D ego-vehicle trajectories from all seeds under both modes and overlay them on a uniform spatial grid. For each grid cell $\mathbf{g}$, we measure the normalized cross-mode behavioral difference for each signal channel k:

$$\Delta_k(\mathbf{g}) = \frac{|\bar{v}_k^{\text{lang}}(\mathbf{g}) - \bar{v}_k(\mathbf{g})|}{\max_{\mathbf{g}'} |\bar{v}_k^{\text{lang}}(\mathbf{g}') - \bar{v}_k(\mathbf{g}')| + \epsilon}, \tag{7}$$

where $\bar{v}_k(\mathbf{g})$ and $\bar{v}_k^{\text{lang}}(\mathbf{g})$ are the seed-averaged behavioral signals at cell $\mathbf{g}$ under direct action mode and language mode, respectively. The behavioral channels include speed, acceleration, heading, and trajectory spread. Additionally, we construct an infraction channel by placing spatial kernels at infraction locations from simulation logs, measuring where the two modes differ in safety outcomes.

We aggregate all channels into a per-cell criticality score $c(\mathbf{g}) = \sum_k w_k \cdot \Delta_k(\mathbf{g})$, threshold the resulting map, and retain the top spatial regions as critical regions $\mathcal{C}_r$. A frame receives $y_{r,t}$=1 only if both y_r=1 and its spatial position falls within $\mathcal{C}_r$ (Eq. 2). During training, we mix route-level and frame-level samples so that the gate learns both

coarse route preference and fine-grained activation patterns. We will fully release the labeling code for complete implementation details.

C.4.3 Temporal Redundancy Cleaning

When the vehicle remains still, such as while waiting at a red light, many consecutive frames contain almost the same scene and therefore produce almost identical hidden states. If we keep all of them, these low-motion moments would be overrepresented during training. We detect such redundant segments by computing cosine similarity between adjacent hidden-state vectors:

$$\mathrm{sim}(\mathbf{h}_t, \mathbf{h}_{t+1}) = \frac{\mathbf{h}_t^\top \mathbf{h}_{t+1}}{\|\mathbf{h}_t\| \cdot \|\mathbf{h}_{t+1}\|} \geq 0.99. \quad (8)$$

Adjacent frames whose similarity exceeds this threshold are treated as one redundant segment of length L. From each segment, we keep only $k = \max(2, \lceil L^\alpha \rceil)$ evenly spaced representative samples, where α=0.5. This sublinear schedule ensures short segments retain at least two samples while preventing long idle periods from overwhelming the dataset. After cleaning, the training set reduces by approximately 15% in frame count while preserving coverage of all driving states.

C.5 Implementation and Hyperparameters

This subsection reports all implementation details needed to reproduce the gate training pipeline.

C.5.1 Data Collection

We extract the hidden state $\mathbf{h} \in \mathbb{R}^d$ from the last transformer layer at the final token position of a prompt-only forward pass, where d=896 is the hidden dimension of InternVL2-1B (Chen et al., 2024b). For both language mode and direct action mode, we forward only the prompt tokens, including visual tokens and system instructions, and collect$\mathbf{h}$ from the same position. This ensures that hidden states from the two modes are aligned and that the gate receives comparable inputs regardless of which mode produced them. All data are collected on approximately 400 routes sampled from the SimLingo training set, stratified by scenario difficulty to cover both common and rare driving situations. For each route, we run both modes separately with 5 random seeds, recording per-frame hidden states and per-route success outcomes. The success rate gap determines training labels as described in Section C.4. BLUE does not require a dedicated data collection effort. In autonomous driving development, validating a model already involves running closed-loop evaluations under different configurations. BLUE simply reuses hidden states and outcomes from these routine runs, making data collection a natural byproduct of the standard pipeline rather than an additional cost.

C.5.2 Gate Architecture and Training

The gate is a single-hidden-layer MLP that maps $\mathbf{h}$ to a scalar activation probability:

$$\mathbf{z} = W_1\mathbf{h} + b_1, \quad p(\mathbf{h}) = \sigma(W_2\,\tilde{\mathbf{z}} + b_2), \quad (9)$$

where $\tilde{\mathbf{z}}$ denotes $\mathbf{z}$ after ReLU activation and dropout, $W_1 \in \mathbb{R}^{m\times d}$, $W_2 \in \mathbb{R}^{1\times m}$, and σ is the sigmoid function. We set hidden dimension m=128 and dropout rate 0.5, yielding 0.11M total trainable parameters. We optimize with Adam using learning rate 0.001, weight decay 0.05, cosine annealing schedule, and batch size 512 for 100 epochs. The training set contains approximately 670K frames.

C.5.3 Computational Cost

The total overhead of BLUE is minimal, largely because its data collection naturally aligns with the standard autonomous driving development workflow. Models must undergo closed-loop road testing before deployment, during which driving outcomes under different configurations are routinely recorded. BLUE simply records hidden states alongside these existing evaluation runs and derives labels automatically by comparing route success rates between the two modes. This means data collection requires minimal additional effort beyond routine validation, no reward engineering, and no human labeling. The raw GPU time consumed by these evaluation runs is approximately 1200 A100 GPU hours, but the vast majority is not a net addition to the development budget. Gate training requires less than 0.1 GPU hours on a single GPU, and the inference overhead is negligible as it adds only a single MLP forward pass per frame. Separately, the language impact analysis in Appendix D.2 consumes approximately ∼2000 A100 GPU hours in total, covering repeated closed-loop experiments across multiple configurations.

C.6 Details of Baseline Gates

This section describes the construction of alternative gating strategies evaluated in Table 7. All baseline gates share the same inference procedure as BLUE: at each frame, the gate produces a binary decision that determines whether to activate language generation or predict actions directly.

Kinematic Gates. We design three rule-based gates, each using a single kinematic feature available at inference time: vehicle speed in m/s, acceleration magnitude as the L2 norm of the 3D acceleration vector in m/s^2, and steering angle as the absolute value of the previous control output normalized to 0–1. A gate activates language generation whenever the corresponding feature exceeds a predefined threshold, selected so that the language activation ratio approximates that of BLUE.

Complexity-based Gate. Rather than relying on frame-level kinematic signals, the complexity-based gate uses route-level scenario complexity as supervision to train a gate. We first compute a composite complexity score for each training route based on structured features extracted from the CARLA scenario configuration file:

$$s = w_1 \cdot f_{\text{scen}} + w_2 \cdot f_{\text{wea}} + w_3 \cdot f_{\text{flow}} + w_4 \cdot f_{\text{freq}}, \quad (10)$$

where f_{scen} reflects the number of sub-scenarios in the route normalized to [0, 1], f_{wea} aggregates fog density, precipitation intensity, and a night-driving indicator, f_{flow} indicates the presence of dynamic traffic flows requiring gap-finding maneuvers, and f_{freq} captures periodic opposing vehicle interactions. We set w_1=0.30, w_2=0.30, w_3=0.25, w_4=0.15. Routes with $s \geq \tau$ are labeled as complex and the rest as simple. All frames within a route share the same label. We then train a gate with the same MLP architecture as BLUE, supervised with these complexity-derived labels. At inference, this gate predicts whether the current frame corresponds to a complex scenario and activates language generation accordingly.

Random Gate. The random gate activates language generation for each frame with a fixed probability, matched to the language activation ratio of BLUE. This baseline isolates the effect of selectively choosing when to generate language from the effect of simply reducing language frequency.

D Additional Experimental Results

The main text reports key results on Bench2Drive. This section provides the complete comparison tables, covering 26 baselines alongside our BLUE on both SimLingo and CriticVLA backbones.

D.1 Full Bench2Drive Comparison

Table 13 presents the full Bench2Drive comparison across 26 methods. For completeness, we include both BLUE (SimLingo) and BLUE (CriticVLA) in a single table. Both variants use only a single front-view camera and language annotations, yet outperform their respective backbones by clear margins. BLUE (SimLingo) achieves the overall best SR and DS among all methods, while BLUE (CriticVLA) also surpasses its backbone and ranks among the top entries. This confirms that the on-demand language use strategy is effective across different VLA architectures under the same evaluation protocol.

D.2 Language Impact Analysis

This section details the experimental settings and statistical methods used to categorize each route as language-helpful, language-neutral, or language-harmful, as summarized in Table 9 of the main text. All analyses in this section are conducted on the Bench2Drive evaluation set. The gate training uses only training routes with no overlap with the evaluation set, as detailed in Appendix C.1.

D.2.1 Experimental Settings

We evaluate the language impact under four settings that vary annotation granularity, annotation language, and VLA backbone. All settings are evaluated on the full 220 routes of Bench2Drive (Jia et al., 2024), running each route with and without language generation across multiple random seeds. We classify each route using two complementary methods: a threshold method based on effect size and a binomial sign test for statistical significance. As Table 14 shows, both methods produce consistent categorizations across all four settings.

SimLingo. The default setting uses the official SimLingo (Renz et al., 2025) checkpoint trained with English annotations at normal granularity. The annotations describe surrounding objects, traffic conditions, and intended maneuvers at each frame.

Brief. To test whether annotation granularity affects the language impact distribution, we retrain SimLingo with brief English annotations that retain only action-relevant instructions and remove detailed scene descriptions. The model architecture, training procedure, and evaluation protocol are identical to the default SimLingo setting.

Chinese. To further test whether the pattern holds under additional annotation settings, we retrain SimLingo with Chinese annotations translated from the default English version. The translated annotations preserve the same content and structure. All other training details remain the same.

Method	Details					Metrics	
	Expert	Camera	LiDAR	Labels	T-Param.	SR (%) ↑	DS ↑
TCP* (Wu et al., 2022)	Think2Drive	3×	×	-	≈26 M	15.00	40.70
TCP-traj* (Wu et al., 2022)	Think2Drive	3×	×	-	≈26 M	30.00	59.90
VAD (Jiang et al., 2023)	Think2Drive	6×	×	O,M	≥25 M	15.00	42.35
UniAD-Base (Hu et al., 2023b)	Think2Drive	6×	×	O,M,S	≥59 M	16.36	45.81
ThinkTwice* (Jia et al., 2023b)	Think2Drive	6×	×	S,D	≈120 M	31.23	62.44
DriveAdaptor* (Jia et al., 2023a)	Think2Drive	6×	×	M,S,D	≈135 M	33.08	64.22
GenAD (Zheng et al., 2024)	Think2Drive	6×	×	O,M	≥25 M	15.90	44.81
TF++ (Zimmerlin et al., 2024)	PDM-Lite	1×	✓	O,M,S,D	≥39 M	67.27	84.21
MomAD (Song et al., 2025)	Think2Drive	6×	×	O,M	≥25 M	18.11	47.91
DriveTrans (Jia et al., 2025)	Think2Drive	6×	×	O,M	≈646 M	35.01	63.46
ETA (Hamdan et al., 2025)	Think2Drive	1×	×	-	≥300 M	48.33	74.33
Hydra-NeXt (Li et al., 2025d)	Think2Drive	2×	×	-	≥25 M	50.00	73.86
Raw2Drive (Yang et al., 2026b)	-	6×	×	M,S	≥25 M	50.24	71.36
DiffusionDrive (Liao et al., 2025)	-	3×	✓	O,S	≈60 M	52.72	77.68
ORION (Fu et al., 2025)	Think2Drive	6×	×	O,M,L	≥300 M	54.62	77.74
AutoVLA (Zhou et al., 2026)	PDM-Lite	1×	×	L	≥1.5 B	57.73	78.84
SimLingo (Renz et al., 2025)	PDM-Lite	1×	×	L	≥300 M	67.27	85.07
HiP-AD (Tang et al., 2025)	Think2Drive	6×	×	O,M,D	≈97 M	69.09	86.77
ReCogDrive (Li et al., 2025c)	Think2Drive	6×	×	L	≥2 B	45.45	71.36
GeRo (Yasarla et al., 2026)	Think2Drive	6×	×	O,M,L	≥3 B	60.10	81.90
DeLL (Du et al., 2026)	PDM-Lite	1×	✓	O,S	≥38 M	68.63	86.86
R2SE (Liu et al., 2026a)	PDM-Lite	1×	✓	O,M,S,D	≥39 M	69.54	86.28
AutoMoT (Huang et al., 2026)	PDM-Lite	1×	✓	-	≈1.6 B	70.00	87.34
BevAD (Holtz et al., 2026)	PDM-Lite	6×	×	O	≥25 M	72.73	88.11
CriticVLA (Yang et al., 2026a)	PDM-Lite	1×	×	L	≥300 M	73.33	88.02
TakeVLA (Gao et al., 2026)	PDM-Lite	1×	×	L	≥300 M	73.73	89.72
BLUE (SimLingo)	PDM-Lite	1×	×	L	**0.11 M**	**76.18**$_{\pm 0.64}$	**90.58**$_{\pm 0.12}$
Δ vs. SimLingo	-	-	-	-	-	+8.91	+5.51
BLUE (CriticVLA)	PDM-Lite	1×	×	L	**0.11 M**	76.04$_{\pm 0.38}$	90.37$_{\pm 0.14}$
Δ vs. CriticVLA	-	-	-	-	-	+2.71	+2.35

Table 13: Full results on Bench2Drive, showing the complete comparison between BLUE and 26 baselines. BLUE (SimLingo) ranks first and BLUE (CriticVLA) ranks second in terms of closed-loop success rate (SR) and driving score (DS). T-Param. reports driving-task trainable parameters; we use published values (≈) where available and conservative lower bounds (≥) derived from the minimum size of trained components. Both BLUE variants train only a 0.11 M gate while keeping the entire VLA backbone frozen, yet surpass methods that employ multi-camera setups, LiDAR, or dense auxiliary labels (O: 3D object detection, M: map, S: semantic segmentation, D: depth, L: language), using only a single front-view camera with language annotations.

CriticVLA. To test whether the pattern generalizes across VLA architectures, we replace the backbone with CriticVLA (Yang et al., 2026a). Unlike SimLingo, which generates language reasoning before predicting actions in a single pass, CriticVLA first produces an initial trajectory and then uses language-based critique to refine the plan.

D.2.2 Statistical Methods

We apply two complementary methods to classify each route. The first provides an intuitive effect-size criterion, and the second offers formal statistical validation.

Threshold method. For each route r, we compute the cross-seed average success rate under language mode $\overline{\mathrm{SR}}^{(r)}_{\text{lang}}$ and under direct action mode $\overline{\mathrm{SR}}^{(r)}_{\text{direct}}$, and take their difference $\Delta^{(r)} = \overline{\mathrm{SR}}^{(r)}_{\text{lang}} - \overline{\mathrm{SR}}^{(r)}_{\text{direct}}$. A route is classified as language-helpful if $\Delta > \tau$, language-harmful if $\Delta < -\tau$, and language-neutral otherwise, where τ=0.1 requires the success rate gap to exceed 10% for a route to be considered meaningfully affected.

Category	Setting	Method	Helpful (%)	Neutral (%)	Harmful (%)
Original	SimLingo	Threshold	14.5	61.8	23.6
		Sign test	14.5	62.7	22.7
Granularity	Brief	Threshold	21.8	52.7	25.5
		Sign test	21.8	52.7	25.5
Language	Chinese	Threshold	18.6	58.2	23.2
		Sign test	15.9	67.7	16.4
Model	CriticVLA	Threshold	19.5	52.3	28.2
		Sign test	18.2	55.5	26.4

Table 14: Language impact categorization across four experimental settings, each evaluated with both the threshold method and the sign test. The two methods yield closely aligned results. Across all settings, the majority of routes are neutral, and language-harmful routes consistently equal or outnumber language-helpful routes.

Sign test. For each route r, we construct all cross-seed pairings between language seeds and direct action seeds, yielding $n_{\text{lang}} \times n_{\text{direct}}$ pairs per route. Among these, we count the discordant pairs: n_+ pairs where language succeeds and direct fails, and n_- pairs where language fails and direct succeeds. Concordant pairs are excluded. Under the null hypothesis that the two modes are equivalent, n_+ follows a Binomial$(n_+ + n_-, 0.5)$ distribution. We apply a one-sided binomial exact test at significance level α=0.15: a route is classified as language-helpful if $P(X \geq n_+) < \alpha$, language-harmful if $P(X \geq n_-) < \alpha$, and language-neutral otherwise. Routes with zero discordant pairs are classified as neutral. We do not apply multiple testing correction because our conclusion depends on the aggregate proportion of the three categories across all routes, not on the classification of any single route.

D.2.3 Detailed Results

Table 14 reports the classification results under both methods across all four settings. The two methods produce highly consistent categorizations. In all settings, the majority of routes fall into the neutral category, and the number of language-harmful routes consistently equals or exceeds the number of language-helpful routes. This pattern holds regardless of annotation granularity, annotation language, or VLA backbone, confirming that the complementarity between the two modes is a general property rather than an artifact of a specific configuration. The agreement between the two methods confirms that our categorization is robust to the choice of statistical criterion, and that selective language activation is warranted across diverse configurations.

Per-scenario results. Beyond aggregate route counts, Figure 8 shows how language impact is distributed across scenario categories under the four settings. The scenario-level view reveals that helpful and harmful effects are not confined to a single scenario family. Many categories are dominated by neutral routes, while language-sensitive cases appear as scattered helpful and harmful groups whose locations can change with the setting. Table 15 maps each scenario index in the figure to its Bench2Drive scenario name. These results support the main conclusion that language generation should be selected on demand rather than used by default at every frame.

D.2.4 Discussion

The cross-setting results above reveal two practical implications for gate design: why each backbone requires its own gate, and why scenario complexity alone cannot replace hidden-state conditioning.

Why each backbone needs its own gate. Figure 8 shows that the helpful, neutral, and harmful distributions change across settings, with a clear shift when moving from SimLingo to CriticVLA. If language utility were determined only by the route, the same scenario index would show similar patterns across backbones. Instead, the pattern changes, indicating that language utility also depends on how each backbone uses language and represents the scene before acting. A gate trained for one backbone therefore learns a readout tied to that model, rather than a universal rule. BLUE trains a separate gate for each backbone, but this adds little cost. As discussed in Section 6, the backbone remains frozen, the gate is a lightweight MLP, and labels come from routine evaluations.

Why scenario complexity is insufficient. Figure 8 also explains why a gate based only on scenario complexity cannot work reliably. All four

settings share the same set of driving scenarios, yet where language helps or hurts shifts when the model or annotation changes. This means the driving data alone, including route type and scenario complexity, cannot determine whether language will help. Recent adaptive VLA methods, including AutoVLA and AdaThinkDrive (Zhou et al., 2026; Luo et al., 2025b), often reduce reasoning in straightforward scenes and keep more reasoning in difficult ones. Our results show that this heuristic misses the key signal: language generation may help on a given scenario under one model but hurt under another. Complexity alone is insufficient to predict when language will help. BLUE therefore conditions the decision on model hidden states rather than on scenario complexity alone.

E Additional Analysis

This section provides additional analysis on the language content generated by SimLingo.

E.1 Language Content Visualization

We visualize the language content generated during closed-loop driving through word clouds, providing readers with an intuitive impression of the vocabulary produced by language generation. Figure 9 presents the word clouds of generated language, where the left panel corresponds to the original SimLingo and the right panel corresponds to BLUE with language generated when the gate activates.

F Additional Statements

We provide statements on artifact licensing, potential risks, broader impact, and the use of large language models in preparing this work.

Licenses and Terms of Use We evaluate BLUE on four benchmarks. Bench2Drive, Longest6 v2, and Fail2Drive are built on the CARLA simulator (Dosovitskiy et al., 2017), which is released under the MIT License. NAVSIM (Dauner et al., 2024) uses real-world driving data and is released under the Apache-2.0 License. All four benchmarks, as well as the SimLingo (Renz et al., 2025) and ReCogDrive (Li et al., 2025c) backbones, are publicly available for academic research. We will fully open-source our code, trained gate checkpoints, training data, and evaluation logs to support reproducibility and future research. Users who build upon our released materials should comply with the licenses of the underlying components and cite the relevant works accordingly.

Potential Risks BLUE is evaluated through closed-loop CARLA simulation and an offline NAVSIM analysis on real-world driving data. The NAVSIM experiment does not involve testing on a physical vehicle. Deployment in real-world conditions would require additional domain-specific training and thorough safety verification to account for sensor noise, distribution shift, and safety-critical edge cases. Any real-world application should follow established engineering protocols for autonomous driving systems.

Broader Impact This work shows that selectively generating language in VLA driving models improves both driving performance and inference efficiency. By reducing unnecessary language generation, BLUE lowers the computational cost of VLA models, making them more practical for real-world vehicle deployment where hardware budgets and energy consumption are constrained. This also reduces the carbon footprint associated with running large language models at inference time. All code, data, and evaluation logs are fully released to facilitate reproducible research on efficient language use in embodied agents. Since BLUE operates through simulation and studies when language benefits driving, we do not foresee direct negative societal consequences from this research.

LLM Use Statement Large language models were used in this work exclusively for English language polishing and assisting with code implementation. All research ideas, experimental designs, analyses, and scientific conclusions are original contributions of the authors. The authors carefully reviewed all LLM-assisted outputs and take full responsibility for the final content of this paper.

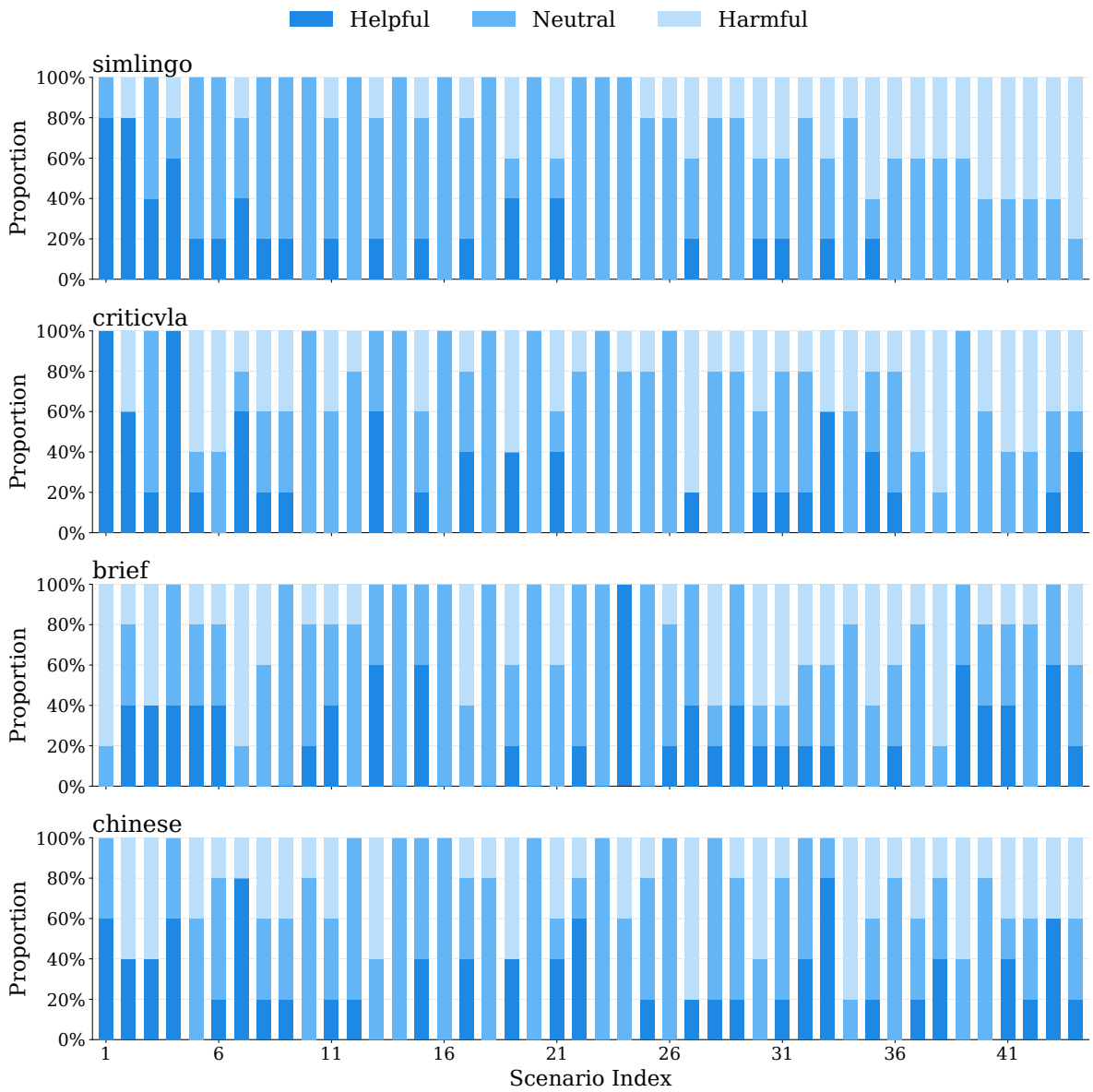

Figure 8: Per-scenario distribution of language effects under four settings. Each bar denotes one Bench2Drive scenario category; all four panels share the same scenario order for direct comparison. Colors indicate routes where language generation is helpful, neutral, or harmful. The scenario index mapping is provided in Table 15.

ID	Scenario	ID	Scenario
1	MergerIntoSlowTrafficV2	23	ControlLoss
2	Accident	24	CrossingBicycleFlow
3	HazardAtSideLane	25	HazardAtSideLaneTwoWays
4	InterurbanActorFlow	26	VanillaSignalizedTurnEncounterGreenLight
5	StaticCutIn	27	VanillaNonSignalizedTurnEncounterStopsign
6	ParkingExit	28	MergerIntoSlowTraffic
7	ConstructionObstacleTwoWays	29	T_Junction
8	OppositeVehicleTakingPriority	30	SignalizedJunctionLeftTurnEnterFlow
9	DynamicObjectCrossing	31	SignalizedJunctionRightTurn
10	VanillaNonSignalizedTurn	32	EnterActorFlow
11	VanillaSignalizedTurnEncounterRedLight	33	NonSignalizedJunctionLeftTurn
12	ParkingCutIn	34	ParkedObstacleTwoWays
13	AccidentTwoWays	35	SignalizedJunctionLeftTurn
14	InvadingTurn	36	OppositeVehicleRunningRedLight
15	HighwayExit	37	ParkingCrossingPedestrian
16	InterurbanAdvancedActorFlow	38	SequentialLaneChange
17	NonSignalizedJunctionLeftTurnEnterFlow	39	VehicleTurningRoutePedestrian
18	YieldToEmergencyVehicle	40	PedestrianCrossing
19	ConstructionObstacle	41	BlockedIntersection
20	HighwayCutIn	42	VehicleOpensDoorTwoWays
21	NonSignalizedJunctionRightTurn	43	VehicleTurningRoute
22	HardBreakRoute	44	ParkedObstacle

Table 15: Mapping between scenario IDs in Figure 8 and Bench2Drive scenario names.

Figure 9: N-gram phrase clouds of generated language during closed-loop driving. Left: SimLingo generates language at every frame. Right: BLUE generates language only when the gate activates. Both panels show the most frequent meaningful phrases (2–5 words) extracted from all evaluation routes.