Scene-Adaptive VLA: Efficient Autonomous Driving via Dynamic Layer Routing

Yujia Yang **Quan Yuan*** **Jiawei Fan** **Yang Li**
Tiange Fu **Xiaoyuan Fu** **Guiyang Luo** **Jinglin Li**
State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications
{yangyj2022,yuanquan,fjw208,leeyang866,tgfu,fuxiaoyuan,luoguiyang,jlli}@bupt.edu.cn

Abstract

Vision-Language-Action (VLA) models offer a promising paradigm for autonomous driving. However, the massive parameter scale of VLAs strains onboard computational resources during real-time inference. This directly increases energy consumption and limits vehicle battery range, making it critical to minimize computational overhead without compromising driving capability and explainability. Current research on efficient VLA-based driving focuses on sequential adaptation (*e.g.*, Chain-of-Thought and token adjustment), leaving structural adaptation underexplored despite parameter redundancy across dynamic driving scenes. To address this gap, we propose Scene-Adaptive VLA, an efficient framework leveraging dynamic layer routing to adjust active parameters based on real-time scenes. Inspired by the temporal continuity of scenes and model predictive uncertainty, our approach characterizes model-perceived scene complexity via frame-level temporal scene variations and decision deviations. To uncover latent scene-to-parameter correspondences, we encode this characterization into scene-aware tokens, alongside two specialized queries. A budget-conditioned layer routing mechanism evaluates these refined queries to allocate a budget that determines the active parameter ratio and subsequently selects appropriate layer combinations. The entire framework is optimized end-to-end. Extensive evaluations on the Bench2Drive benchmark show that when adapted to a state-of-the-art VLA model, our approach significantly reduces inference-time computational overhead by 53.5% with negligible degradation in driving and language capabilities. Code is available at `https:github.com/yang-luvu/Scene-Adaptive-VLA`.

1 Introduction

Built on Vision-Language Models (VLMs), Vision-Language-Action (VLA) models have emerged as a dominant paradigm in autonomous driving, unifying visual perception, linguistic reasoning, and action prediction within a cohesive framework [1–4]. However, the massive parameter scale of VLA models creates severe onboard computational resource pressure for real-time inference, directly increasing energy consumption and reducing vehicle battery range. Consequently, minimizing computational overhead without compromising driving capability and explainability has become a paramount objective in VLA-based autonomous driving research [5, 6]. Currently, efforts to improve inference efficiency in VLMs can be broadly categorized into two directions: *sequential adaptation*, which dynamically adjusts the number of input or output tokens based on the task [7–10], and *structural adaptation*, which selectively activates model parameters during inference to adapt to the task [11]. In the domain of VLA-based autonomous driving, existing efficient inference strategies pre-

*Corresponding author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

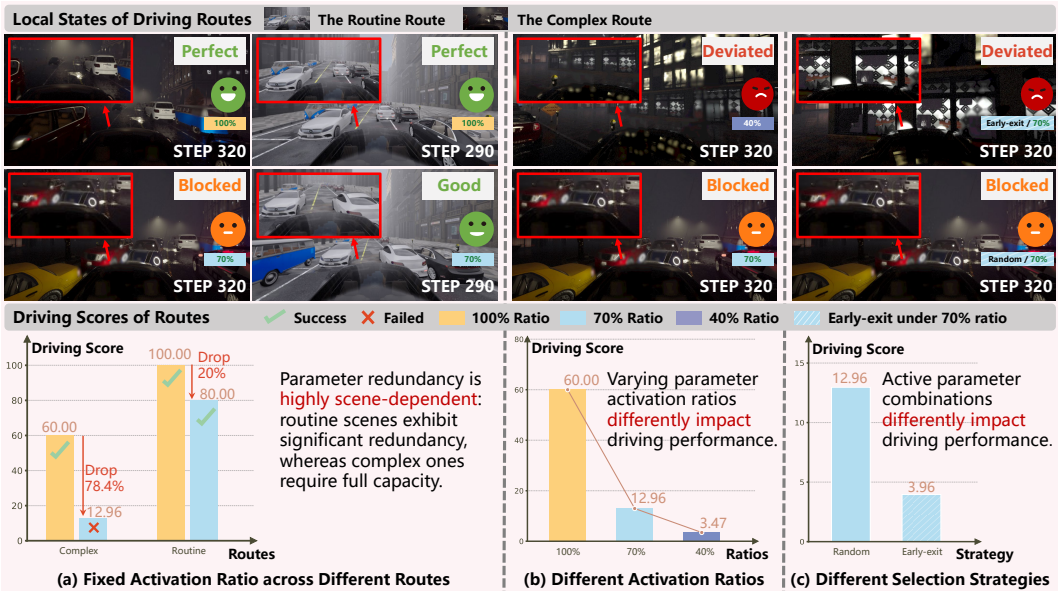


Figure 1: Driving performance under varying capacities of a state-of-the-art model [3] on Bench2Drive. (a) Fixed Ratio Across Routes: The model with 70% active parameters completely fails on routes featuring more complex driving scenes, yet completes routes with only a 20% score drop on routes dominated by routine scenes. (b) Activation Ratios (Complex Route): Reducing parameters severely degrades performance, with lower ratios causing critical local failures (e.g., lane deviation at 40%, blockage at 70%). (c) Layer Selection Strategies (Complex Route): At 70% activation, an early-exit strategy significantly underperforms random layer dropping, proving that active parameter combinations are critical.

dominantly focus on sequential adaptation, such as adjusting the length of Chain-of-Thought (CoT) reasoning [5] or modulating input token counts based on the scene [6]. Conversely, structural adaptation remains largely underexplored.

As illustrated in Fig. 1(a), the ratio of parameters required for successful driving varies significantly across scenes of varying complexity. Structural adaptation thus holds substantial potential to enable dynamic, task-aware adjustment of active parameters, thereby facilitating efficient autonomous driving. Furthermore, varying the active parameter ratio and combination yields distinctly different performance within identical scenes as shown in Fig. 1(b) and (c). Therefore, we aim to design a VLA framework that dynamically determines the optimal parameter ratio and combination for each driving frame. However, establishing this scene-to-parameter mapping is non-trivial due to the considerable diversity of driving scenes and the abstract nature of model capabilities. Specifically, determining the *scene-to-parameter ratio* is challenging because perceiving scene complexity and translating it into a measure of required model capacity is a highly implicit process. Furthermore, optimizing the *scene-to-parameter combination* is complex. Accurately selecting optimal active parameters for a specific driving scene is complicated by the entangled black-box nature of neural networks and the high variance of driving scenarios.

To address these challenges, we propose Scene-Adaptive VLA, an efficient VLA-based autonomous driving framework leveraging dynamic layer routing. Inspired by the temporal continuity of driving scenes and model predictive uncertainty theory [12–14], Scene-Adaptive VLA characterizes model-perceived scene complexity via frame-level temporal scene variations and decision deviations. To uncover latent scene-to-parameter correspondences, we formulate scene-aware tokens that encode the aforementioned scene characterization, alongside two specialized queries: a budget token for determining parameter ratios and a routing token for selecting layer combinations. After processing these tokens through shallow layers of the Large Language Model (LLM) to capture their complex interdependencies, we introduce a budget-conditioned layer routing mechanism. This mechanism evaluates the refined queries to first determine the active parameter ratio and subsequently route inputs to the optimal layer combinations. Finally, the entire framework is optimized via two-stage fine-tuning supervised by both driving performance and resource consumption, effectively establishing an efficient scene-to-parameter mapping. Theoretically, our framework can be integrated into

prevailing VLA architectures that employ dense backbones. To validate its effectiveness under hardware constraints, we adapt our framework to SimLingo [3], which achieves optimal performance with minimal parameter scale in VLA-based driving models. The main contributions of this work are summarized as follows:

1. A novel formulation for the characterization of model-perceived scene complexity. Leveraging temporal continuity of driving scenes and model predictive uncertainty theory, it effectively characterizes scene complexity via temporal frame-level driving scene variations and decision deviations, encoding them into scene-aware tokens.

2. A lightweight, budget-conditioned routing mechanism for efficient dynamic parameter activation. Utilizing two queries alongside the extracted scene-aware tokens and the LLM's shallow layers, this mechanism dynamically allocates a real-time inference budget to determine the active parameter ratio and selects the optimal parameter combination via layer routing, all optimized end-to-end.

3. Exceptional closed-loop performance on the Bench2Drive benchmark. When adapted to the state-of-the-art (SOTA) VLA-based driving model, our framework achieves a 53.5% reduction in inference-time computational overhead with negligible degradation in driving proficiency and language capabilities, as demonstrated by extensive evaluations on the Bench2Drive benchmark and language tasks.

2 Related Work

VLA-based End-to-end Autonomous Driving. End-to-end autonomous driving has witnessed rapid advancements in recent years. These frameworks integrate perception and planning within unified architectures to enhance driving efficacy [15, 16]. To address complex long-tail scenarios, recent approaches integrate VLMs either for language-centric scenario understanding [17, 2] or for guiding downstream driving models [18, 19]. However, these decoupled paradigms lack direct end-to-end optimization grounded in continuous actions. Consequently, VLA architectures have emerged to directly couple VLMs with action generation. While Drivegpt4-v2 [1] maps multimodal inputs to control signals and EMMA [20] unifies tasks via plain text representations, a critical modality gap remains. To bridge this gap between semantic reasoning and numerical control, ORION [4] incorporates a generative planner alongside a query-based temporal module, enabling unified optimization for scenario reasoning and trajectory prediction. Concurrently, SimLingo [3] enforces strict language-action alignment via an auxiliary task trained on synthetic instruction-action pairs, yielding superior performance in closed-loop evaluation.

Efficient VLA-based Model Inference in Autonomous Driving. Despite their formidable capabilities, the immense parameter scale of VLA models imposes a prohibitive computational burden on vehicular hardware. To address this, recent studies have explored efficient inference strategies. For instance, AutoVLA [5] reduces reasoning overhead by introducing a fast-and-slow thinking mechanism that dynamically adjusts CoT lengths via Group Relative Policy Optimization [21] (GRPO). Similarly, DriveMoE [6] alleviates the cost of long vision sequences by employing a Mixture-of-Experts architecture with on-demand vision token filtering. While effective, these methods largely overlook active parameter redundancy during inference.

Structural Adaptation in VLMs. Structural adaptation optimizes the performance-efficiency trade-off by dynamically allocating active parameters based on task demands. For Vision Transformers and LLMs, this is typically achieved by selectively dropping computational modules or employing adaptive early-exit mechanisms [22–24]. Extending these paradigms to VLMs, AdaLLaVA [11] introduces a framework that selectively disables entire Transformer layers using binary switches under an inference latency constraint. For structural adaptation in VLA-based autonomous driving, it is essential to determine the optimal parameter activation ratio and enable flexible parameter combinations conditioned on driving scenes. Thus, we adopt a similar paradigm to achieve active-ratio-aware layer selection for VLA models.

3 Scene-Adaptive VLA

To validate our method under hardware constraints, we adapt it to SimLingo [3], a VLA driving model that achieves optimal performance with minimal parameters. As illustrated in Fig. 2, our

approach enables scene-adaptive parameter activation through two core components. First, Model-Perceived Complexity Characterization encodes scene complexity into scene-aware tokens by quantifying temporal visual variations and inter-frame decision deviations. Second, Budget-Conditioned Layer Routing dynamically allocates a real-time computational budget to dictate the active parameter ratio and layer combinations. These decisions are governed by budget and layer routing queries, which are processed in the shallow layers of the LLM alongside scene-aware tokens to capture scene-parameter correlations. The training proceeds in two stages. Stage 1 employs SFT with randomized budgets to initialize scene and routing representations. Stage 2 leverages GRPO [21] for end-to-end optimization, utilizing a composite reward derived from driving proficiency and computational overhead.

3.1 Framework

Base VLA Model. The backbone is InternVL2-1B [25], comprising the InternViT-300M vision encoder and the Qwen2-0.5B-Instruct [26] LLM. To ensure lightweight and efficient execution, we partition the language model into L_{pre} shallow prefix layers and L_{sub} subsequent routable layers. We augment this architecture with two compact multi-layer perceptron (MLP) schedulers. Operating on intermediate features extracted by the prefix layers, these schedulers leverage the infused scene-aware tokens and specialized queries to dynamically determine the routing plan for the remaining layers without introducing substantial computational overhead.

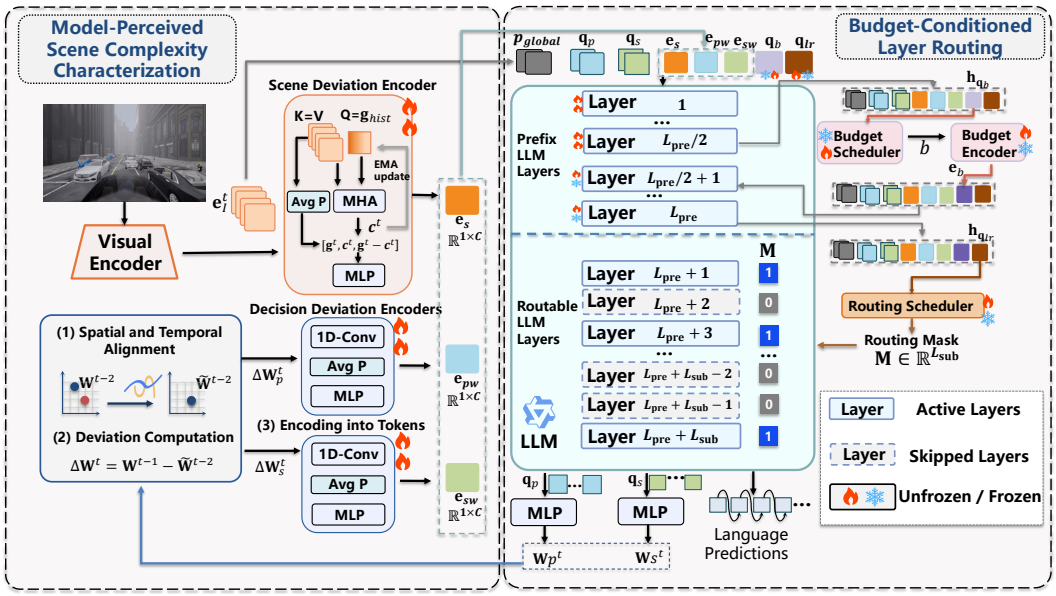


Figure 2: Overview of the Scene-Adaptive VLA, featuring Model-Perceived Scene Complexity Characterization and Budget-Conditioned Layer Routing. Adjacent icons denote paramter status (left/top: Stage 1; right/bottom: Stage 2). **Left**: The characterization module encodes temporal visual variations and inter-frame decision deviations into scene-aware tokens ($\mathbf{e}_s$, $\mathbf{e}_{pw}$, $\mathbf{e}_{sw}$). **Right**: The LLM prefix layers process these tokens alongside budget ($\mathbf{q}_b$) and routing ($\mathbf{q}_{lr}$) queries. A budget scheduler allocates a real-time computational budget b based on intermediate $\mathbf{q}_b$ features from the middle prefix layers. At the final prefix layer, a routing scheduler utilizes $\mathbf{q}_{lr}$ to generate a binary mask $\mathbf{M}$, dynamically activating specific layers and bypassing others via residual connections to yield final predictions.

Input Representation. Following SimLingo, the driving-specific inputs comprise the visual scene tokens $\mathbf{e}_I \in \mathbb{R}^{N_c \times C}$, the task prompt p_{task}, navigation tokens $\mathbf{e}_{\text{nav}}$, ego-vehicle speed v, path waypoint queries $\mathbf{q}_p \in \mathbb{R}^{N_p \times C}$ and speed waypoint queries $\mathbf{q}_s \in \mathbb{R}^{N_s \times C}$, where N_c, N_p, and N_s denote the respective token counts, and C represents the LLM hidden dimension. The elements form a global prompt p_{global}: `<image features` $\mathbf{e}_I$`>\n Current speed:` `<`v`> m/s. Command:` `<nav. features` $\mathbf{e}_{\text{nav}}$`>.` `<task prompt` p_{task}`>`. To facilitate scene-adaptive layer routing, we introduce five additional tokens in $\mathbb{R}^{1\times C}$. The scene-aware inputs comprise a scene deviation token

$\mathbf{e}_s$ alongside decision deviation tokens for path ($\mathbf{e}_{pw}$) and speed ($\mathbf{e}_{sw}$), with their construction detailed in Section 3.2. Specialized queries consist of the budget query $\mathbf{q}_b$ and the layer routing query $\mathbf{q}_{lr}$, which govern budget allocation and layer routing, respectively. The complete sequence fed into the model is concatenated in order as $I_{\text{llm}} = [p_{\text{global}}, \mathbf{q}_p, \mathbf{q}_s, \mathbf{e}_s, \mathbf{e}_{pw}, \mathbf{e}_{sw}, \mathbf{q}_b, \mathbf{q}_{lr}]$.

3.2 Model-Perceived Scene Complexity Characterization

To realize scene-adaptive parameter activation, quantifying complexity from the model's perspective is essential. Drawing on the principle that prediction variance with similar inputs indicates uncertainty [13, 12], we characterize complexity via two complementary signals. The first signal is temporal scene-level deviations that capture dynamic visual variations. The second signal is inter-frame decision deviations, reflecting model uncertainty through decision discrepancies across adjacent frames. Together, these signals provide a scene-complexity-aware representation to inform downstream budget and layer routing decisions.

Scene Temporal Deviation. Building upon the visual scene tokens $\mathbf{e}_I^t$ extracted by the visual encoder, we obtain a global scene representation at time t, $\mathbf{g}^t$, via mean pooling: $\mathbf{g}^t = \frac{1}{N_c}\sum_{i=1}^{N_c} \mathbf{e}_I^{t,i}$, where $\mathbf{e}_I^{t,i} \in \mathbb{R}^C$. To capture temporal variations, we maintain a historical scene representation $\mathbf{g}_{\text{hist}}$, updated across frames via an exponential moving average (EMA). A multi-head cross-attention (MHA) mechanism explicitly models temporal evolution by:

$$\mathbf{c}^t = \text{MHA}(\text{query} = \mathbf{g}_{\text{hist}}, \text{key} = \mathbf{e}_I^t, \text{value} = \mathbf{e}_I^t), \mathbf{g}_{\text{hist}} = \alpha \mathbf{c}^t + (1-\alpha)\mathbf{g}_{\text{hist}}, \tag{1}$$

where $\alpha \in [0, 1]$ controls the update rate and $\mathbf{c}^t$ captures deviations from the expected visual state. The scene deviation token $\mathbf{e}_s$ is then computed by concatenating features along the channel dimension:

$$\mathbf{e}_s = \text{MLP}_s([\mathbf{g}^t, \mathbf{c}^t, \mathbf{g}^t - \mathbf{c}^t]) \in \mathbb{R}^{1\times C}. \tag{2}$$

This design encodes both the absolute scene content and its historical deviation into a single compact token.

Decision Deviation. To maintain an efficient single-forward-pass inference paradigm, we evaluate the decision deviation of the current timestep t by leveraging cached outputs from the two preceding timesteps, $t-1$ and $t-2$. This is feasible due to the substantial temporal overlap between consecutive long-horizon predicted trajectories. Let $\mathbf{W}_p^{t-1}$ and $\mathbf{W}_p^{t-2}$ denote the predicted cumulative-sum path waypoints from the previous two frames. After adjusting $\mathbf{W}_p^{t-2}$ for ego-motion via relative translation and yaw, we apply linear interpolation for temporal alignment to obtain the aligned past trajectory $\tilde{\mathbf{W}}_p^{t-2}$. The per-waypoint path deviation is $\Delta\mathbf{W}_p = \mathbf{W}_p^{t-1} - \tilde{\mathbf{W}}_p^{t-2}$. This sequence is condensed into a unified path deviation token $\mathbf{e}_{pw}$ via:

$$\mathbf{e}_{pw} = \text{MLP}_{pw}(\text{AvgPooling}(\text{Conv1D}_{pw}(\Delta\mathbf{W}_p))) \in \mathbb{R}^{1\times C}. \tag{3}$$

Similarly, for the predicted speed waypoints $\mathbf{W}_s \in \mathbb{R}^{N_s\times 2}$, we apply identical spatial-temporal alignment to obtain $\tilde{\mathbf{W}}_s^{t-2}$. By measuring the Euclidean distance between consecutive waypoints, we derive the sequential speed profiles $\mathbf{v}^{t-1}$ and $\tilde{\mathbf{v}}^{t-2} \in \mathbb{R}^{(N_s-1)\times 1}$. The speed deviation $\Delta\mathbf{v} = \mathbf{v}^{t-1} - \tilde{\mathbf{v}}^{t-2}$ is encoded into $\mathbf{e}_{sw} \in \mathbb{R}^{1\times C}$ using an analogous network:

$$\mathbf{e}_{sw} = \text{MLP}_{sw}(\text{AvgPooling}(\text{Conv1D}_{sw}(\Delta\mathbf{v}))) \in \mathbb{R}^{1\times C}. \tag{4}$$

By concatenating these scene-aware tokens directly into the input sequence, the shallow prefix layers can deeply fuse them with the intrinsic scene features, establishing the foundation for the budget-conditioned layer routing detailed in Section 3.3.

3.3 Budget-Conditioned Layer Routing Mechanism

To uncover latent scene-to-parameter correspondences, we introduce a budget-conditioned layer routing mechanism. First, a budget allocation module determines the active parameter ratio via the budget query. Subsequently, a routing module selects optimal layer combinations guided by the layer routing query under the allocated budget. The framework is then optimized through two-stage fine-tuning that balances driving performance and computational overhead to establish an efficient scene-to-parameter mapping.

Scene-Adaptive Layer Routing Inference. As illustrated in Fig. 2, midway through the L_{pre} prefix layers, the budget scheduler extracts the hidden feature of the budget query token, $\mathbf{h}_{q_b}$, to select a discrete budget action $a \in \{0, ..., K-1\}$. This selected action is subsequently scaled to determine the assigned budget $b = \max(\frac{1}{K} \cdot (a+1),\, b_{\min})$. The lower bound, $b_{\min}$, guarantees a foundational reasoning capacity for the model and is defined as $b_{\min} = \max(0.2, L_{\text{pre}}/L_{\text{llm}})$, where $L_{\text{llm}} = L_{\text{pre}} + L_{\text{sub}}$. The budget b is then encoded into $\mathbf{e}_b$ via sinusoidal embeddings and a budget encoder f_{enc}, replacing the initial $\mathbf{q}_b$ token. At the prefix boundary, the routing scheduler f_{routing} processes the concatenated layer routing query $\mathbf{q}_{lr}$ and $\mathbf{e}_b$ to generate routing logits $\boldsymbol{\pi}_{\text{layer}} \in \mathbb{R}^{L_{\text{sub}}}$ for the L_{sub} subsequent layers.

To enforce the budget constraint, b is quantized into an active sub-layer count $k = \max(0, \lfloor L_{\text{llm}} \cdot b \rfloor - L_{\text{pre}})$. Then we obtain the binary routing mask $\mathbf{M} \in \{0,1\}^{L_{\text{sub}}}$ via Gumbel-Softmax top-k sampling:

$$\mathbf{M} = \text{GumbelSoftmax}_{\tau}(\boldsymbol{\pi}_{\text{layer}})_{\text{top-}k}, \text{s.t. } \mathbf{M}_j \in \{0,1\},\ \sum_{j=1}^{L_{\text{sub}}} \mathbf{M}_j = k. \tag{5}$$

The straight-through estimator ensures valid gradient flow during training. The mask $\mathbf{M}$ acts as a binary gate for the remaining decoder layers, efficiently bypassing unselected layers via residual connections to decode language tokens and waypoints. To optimize this architecture effectively, we propose a two-stage training pipeline.

Stage 1 Scene-aware Representation and Layer Routing Learning. Stage 1 initializes scene-aware representations and budget-aware routing via SFT. With the budget scheduler frozen, budgets are uniformly sampled from $b \sim \mathcal{U}[b_{\min}, 1]$. We jointly optimize the layer routing query $\mathbf{q}_{lr}$, the scene-aware encoders, the routing scheduler f_{routing}, the budget encoder f_{enc}, and the LLM prefix layers (tuned via LoRA [27]). The primary supervision minimizes the Smooth-L1 waypoint loss $\mathcal{L}_{\text{wp}}$ and the auto-regressive language loss $\mathcal{L}_{\text{lang}}$, following the formulation in SimLingo:

$$\mathcal{L}_{\text{wp}} = \text{SmoothL1}(\mathbf{W}_p, \mathbf{W}_p^{\text{gt}}) + \text{SmoothL1}(\mathbf{W}_s, \mathbf{W}_s^{\text{gt}}), \mathcal{L}_{\text{lang}} = -\sum_{t=1}^{T} \log P(l_t \mid l_{<t}, I_{\text{llm}}). \tag{6}$$

Here $\mathbf{W}_p^{\text{gt}}$, $\mathbf{W}_s^{\text{gt}}$, and l_t denote ground-truth path waypoints, speed waypoints, and language tokens, respectively. To prevent degenerate routing behaviors, we introduce two auxiliary regularizers. The diversity loss $\mathcal{L}_{\text{div}}$ encourages distinct routing patterns for dissimilar scenes. The balance loss $\mathcal{L}_{\text{bal}}$ aligns the empirical per-layer activation rate $\bar{a}_j$ with the target budget ratio $\bar{b}$:

$$\mathcal{L}_{\text{div}} = \frac{1}{B^2} \sum_{i,j} \left[\text{sim}(\mathbf{M}_i, \mathbf{M}_j) - \text{sim}(\mathbf{z}_i, \mathbf{z}_j) - m_{\text{d}}\right]_+^2,\ \mathcal{L}_{\text{bal}} = \frac{1}{L_{\text{sub}}} \sum_j \left[|\bar{a}_j - \bar{b}| - \varepsilon_{\text{b}}\right]_+^2, \tag{7}$$

where $[\cdot]_+$ is the ReLU function, $\text{sim}(\cdot,\cdot)$ is cosine similarity, and m_{d} and ε_{b} are margin hyperparameters. $\mathbf{z}_i$ is the max-pooled scene feature. $\bar{a}_j = \frac{1}{B}\sum_i \mathbf{M}_{i,j}$ represents the mean activation rate of sub-layer j over batch B. The overall Stage 1 objective is the weighted sum: $\mathcal{L}_{\text{stage1}} = \lambda_{\text{wp}}\mathcal{L}_{\text{wp}} + \lambda_{\text{lang}}\mathcal{L}_{\text{lang}} + \lambda_{\text{div}}\mathcal{L}_{\text{div}} + \lambda_{\text{bal}}\mathcal{L}_{\text{bal}}$.

Stage 2 Reinforcement Fine-Tuning for Budget Allocation. To explicitly learn an optimal budget allocation policy, we apply GRPO [21]. We unfreeze the budget scheduler f_{budget} and the budget query $\mathbf{q}_b$ while freezing the routing scheduler f_{routing}, the budget encoder f_{enc}, the layer routing query $\mathbf{q}_{lr}$, and the latter half of the prefix layers. The policy $\pi_\theta(a \mid \mathbf{h}_{q_b}) = \text{Categorical}\big(\text{softmax}(f_{\text{budget}}(\mathbf{h}_{q_b}))\big)$ maps the intermediate representation to a distribution over discrete budget actions. For each sample i, we draw G actions $\{a_{i,g}\}_{g=1}^{G}$ from the old policy π_{old}. The reward $r_{i,g}$ balances driving quality improvements against budget parsimony:

$$r_{i,g} = w_d \tanh\left(\mathcal{L}_{\text{full},i} - \mathcal{L}_{i,g}\right) - w_b \frac{b(a_{i,g}) - b_{\min}}{1 - b_{\min}}, A_{i,g} = \frac{r_{i,g} - \mu_i}{\sigma_i + \varepsilon_a} + w_a r_{i,g}, \tag{8}$$

where $\mathcal{L}_{\text{full},i}$ and $\mathcal{L}_{i,g}$ denote the full-budget reference loss and the loss under action $a_{i,g}$. Weights w_d, w_b, and w_a control the driving-quality, budget-parsimony, and absolute driving-performance terms, respectively. μ_i and σ_i are the mean and standard deviation of rewards across the G groups, and ε_a ensures numerical stability. The policy is updated via a clipped surrogate objective with KL divergence regularization:

$$\mathcal{L}_{\text{policy}} = -\mathbb{E}_{i,g}\Big[\min\left(\rho_{i,g}(\theta)A_{i,g}, \text{clip}(\rho_{i,g}(\theta), 1-\varepsilon_c, 1+\varepsilon_c)A_{i,g}\right)\Big] + \beta \text{KL}(\pi_\theta \parallel \pi_{\text{old}}), \tag{9}$$

where $\rho_{i,g}(\theta) = \frac{\pi_\theta(a_{i,g}|\mathbf{h}_{q_b}^{(i)})}{\pi_{\text{old}}(a_{i,g}|\mathbf{h}_{q_b}^{(i)})}$ is the importance sampling ratio. θ and θ_{old} denote the current and old policy parameters, ε_c and β are hyperparameters controlling the clipping range and the weight of the KL. To prevent catastrophic deviation in the unfrozen parameters of the prefix layers' LoRA and scene-aware encoders, we introduce an anchor loss: $\mathcal{L}_{\text{anchor}} = \frac{1}{|\boldsymbol{\phi}|}\sum_j(\phi_j - \bar{\phi}_j)^2$. Here, $\boldsymbol{\phi}$ represents the vector of currently trainable parameters, and $\bar{\boldsymbol{\phi}}$ denotes their corresponding frozen values obtained from Stage 1. The total Stage 2 objective is therefore: $\mathcal{L}_{\text{stage2}} = \mathcal{L}_{\text{policy}} + \beta_{\text{anchor}}\mathcal{L}_{\text{anchor}}$.

4 Experiments

4.1 Benchmarks and Metrics

Inference Computational Overhead. To demonstrate the efficiency of our approach, we report the estimated per-decision floating-point operations (FLOPs) alongside measured runtime metrics, including per-decision latency and peak memory. We compute FLOPs for the LLM component only, which dominates the computation of these VLA models. All runtime metrics are measured on Bench2Drive under identical settings using an RTX 4090. Language generation is enabled for all methods except DriveMoE, which does not support language output. Since all compared methods use dense decoder-only LLM backbones, we estimate LLM computation as $\text{FLOPs}_{\text{LLM}} \approx 2P(N_{\text{in}} + N_{\text{out}})$ [28, 29], where P is average number of active LLM parameters, and N_{in} and N_{out} denote the average number of input and output tokens, respectively. Detailed token configurations are provided in Appendix A. For baselines with dynamic computation, we report their lowest-compute settings, i.e., AutoVLA in fast-thinking mode and DriveMoE with single-view input.

Table 1: LLM computation overhead per driving decision.

Method	Avg. P(B)	N_{in}	N_{out}	TFLOPs ↓	Latency (ms) ↓	Mem. (GB) ↓	Language
DriveMoE[6]	3.00	527	1	3.17	<u>319.54</u>	14.85	×
AutoVLA[5]	3.00	1736	18	10.52	1322.85	7.18	✓
ORION[4]	7.00	538	110	9.07	5440.70	17.83	✓
ORION+SA	3.68	538	110	4.77	2971.60	17.83	✓
SimLingo[3]	0.50	569	22	<u>0.59</u>	410.79	<u>2.03</u>	✓
SimLingo+SA	0.23	569–576	22	**0.27**	**214.01**	**2.01**	✓

Bench2Drive. We evaluate closed-loop performance on the Bench2Drive benchmark [30] based on the CARLA simulator, comprising 220 diverse 150-meter routes. We report four primary official metrics: Driving Score, Success Rate, Efficiency, and Comfortness. Success rate measures the proportion of routes successfully completed within the time limit, while driving score incorporates both route completion status and violation penalties. Efficiency and Comfortness evaluate the autonomous driving system's speed performance and passenger comfort during navigation, respectively.

Driving Linguistic Capabilities Test. To assess the impact of our approach on the base VLA model's language abilities, we evaluate its performance on the linguistic benchmarks introduced by SimLingo using SPICE [31] as the metric. Specifically, we utilize the DriveLM-hard visual question answering (VQA) and the driving Commentary validation set. This dataset comprises 4,355 VQA test samples and 4,678 driving commentary samples, which primarily focus on interpreting the current driving frame. Consequently, the VQA subset effectively tests general scene comprehension, while the Commentary subset systematically measures the model's capacity to articulate driving decisions and their underlying rationales in natural language.

4.2 Implementation Details

Model Setting. We integrate Scene-Adaptive VLA into SimLingo, denoted as SimLingo+SA. We train our model using the dataset provided by SimLingo, jointly optimizing for driving and language tasks on two NVIDIA RTX 4090 GPUs. Based on the LLM architecture, we set the number of

Table 2: Comparison with SOTA driving methods on the closed-loop benchmark.

Method	DS↑	SR(%) ↑	Efficiency↑	Comfortness↑	TFLOPs↓
UniAD-Base[15]	45.81	16.36	129.21	43.58	3.42
VAD[16]	42.35	15.00	157.94	46.01	1.14
TCP-traj[32]	59.90	30.00	76.54	18.08	**0.04**
AutoVLA[5]	78.84	57.73	146.93	39.33	10.52
DriveMoE[6]	74.22	48.64	175.96	15.31	3.17
ORION[4]	77.74	54.62	151.48	17.38	9.07
ORION+SA@100%	77.45	53.63	153.03	21.38	9.07
ORION+SA@65%	75.07	49.54	152.47	22.88	5.90
ORION+SA	75.88	50.45	150.09	25.29	4.77
SimLingo[3]	85.07	**67.27**	259.23	33.67	0.59
SimLingo+SA	84.87	<u>65.45</u>	**266.19**	**54.36**	0.27
SimLingo+SA@100%	**86.83**	**67.27**	<u>262.13</u>	44.79	0.59
SimLingo+SA@70%	<u>85.44</u>	64.09	259.06	53.19	0.41
SimLingo+SA@40%	78.20	59.09	262.05	<u>53.20</u>	<u>0.24</u>

prefix layers to $L_{\text{pre}} = 4$ and the routable sub-layers to $L_{\text{sub}} = 20$. During training, LoRA is applied to the prefix layers to enhance efficiency. In Stage 1, we use a learning rate of 3e-5 and a batch size of 6, training for one epoch (approximately 90k steps). The loss weights for waypoints and language modeling are set to $\lambda_{\text{wp}} = \lambda_{\text{lang}} = 1.0$, with diversity and balance regularization weights $\lambda_{\text{div}} = 0.05$ and $\lambda_{\text{bal}} = 0.01$, respectively. In Stage 2, the learning rate is adjusted to 1e-5 with a batch size of 4, training for roughly 40k steps. GRPO is configured with a group size of $G = 4$ and performs three policy updates per step; the weights for driving performance, budget, and absolute driving performance are $w_d = 1.0, w_b = 0.05, w_a = 0.3$.

4.3 Main Results

Inference Computational Overhead. We evaluate the average LLM computational overhead per driving decision across VLA models using FLOPs and measured runtime metrics. In all tables within this section, the best results are highlighted in bold, and the second-best results are underlined. As Tab. 1 shows, our framework reduces SimLingo's FLOPs by 53.5% and latency by 47.9% via scene-adaptive dynamic parameter activation. For comparison, we further evaluate SimLingo+SA against efficient VLA baselines, including DriveMoE and AutoVLA. Despite disabling language generation, DriveMoE is still slower and uses 7.4× more peak memory, while AutoVLA incurs substantially higher latency even in fast-thinking mode. In addition, our lightweight framework design contributes significantly to this efficiency: it introduces auxiliary modules comprising a mere 14M parameters, and is specifically designed to exclusively process the 5 additional input tokens through the initial prefix layers. This incurs virtually no extra computational overhead due to the shallow depth of these layers.

Closed-loop Driving Performance. We evaluate the closed-loop driving performance of SimLingo+SA. The results are averaged over three random seeds. As demonstrated in Tab. 2, our method significantly reduces the computational overhead of the original SimLingo framework by 53.5% while incurring only a marginal performance degradation, specifically a 0.2% drop in DS and a 1.82% drop in SR.

Table 3: Language ability evaluation.

Method	Budget	VQA↑	Commentary↑
InternVL2-2B	-	44.40	9.04
SimLingo	-	**56.77**	**38.04**
SimLingo+SA	100%	<u>54.91</u>	<u>35.37</u>
SimLingo+SA	70%	53.55	35.28

This demonstrates that our approach can dynamically select the appropriate parameter ratio and combination conditioned on the driving scene, thereby substantially reducing the runtime resource usage of VLA models. Furthermore, by effectively capturing inter-frame dynamics and decision deviations, our model surpasses SimLingo in Efficiency and notably enhances Comfortness, achieving the best results. Additionally, we evaluate the driving performance of our model under fixed budgets, denoted as SimLingo+SA@b. Under a full budget, our framework improves all other driv-

ing metrics while maintaining the Success Rate, demonstrating that our approach enables dynamic scene-adaptive parameter adjustment while improving the foundational model's performance ceiling. The multi-ability results are shown in Appendix B.

Linguistic Capabilities. To verify the impact of our method on linguistic reasoning capabilities, we evaluate our model on the driving VQA dataset introduced by SimLingo. As shown in Tab. 3, although our framework incurs a slight performance degradation compared to the original model, specifically a 1.86% drop in DriveLM-VQA and a 2.67% drop in Commentary under full budget, it effectively preserves the explainability of driving scenes and decisions, consistently outperforming InternVL2-2B under different budgets.



Figure 3: Visualization of scene-adaptive parameter activation. Left: driving scenes at selected steps. Right: corresponding budget and sampled layer combinations.

Qualitative Analysis. To illustrate the dynamic parameter activation capability of our Scene-Adaptive VLA, we visualize selected driving frames from Route 3 of the Bench2Drive benchmark as shown in Fig. 3. Scenes between driving steps 50 and 80 are relatively routine, requiring ego-vehicle merely to follow the lane; thus, an average of only 52% of the parameters is activated. As the vehicle progressively approaches a dense traffic area, the allocated inference budget gradually increases. Subsequently, during steps 90 to 120, a slow-moving cyclist appears ahead, necessitating a lane change to maintain a safe distance. Consequently, a higher proportion of parameters is dynamically allocated to execute this complex maneuver effectively.

4.4 Ablation Studies

Scene-aware Characterization. To validate the efficacy of our scene complexity characterization, we remove the scene-deviation, speed-deviation, and path-deviation tokens individually or jointly and compare the variants with the full model. As shown in Tab. 4, the full model achieves the highest DS and SR with the lowest average budget. Removing any deviation token impairs complexity estimation and budget allocation, leading to higher budgets and degraded performance. Removing the scene-deviation token only slightly increases the budget to 54.37%, likely because visual tokens still retain scene-level information through prefix-layer fusion. In contrast, removing the speed-deviation or path-deviation token increases the budget to 71.89% and 66.70%, respectively, indicating that explicit historical decision uncertainty is critical for complexity assessment. When all deviation tokens are removed, the model requires the highest budget of 73.59% while achieving lower DS and SR, further highlighting the complementary roles of three feature tokens encoding different deviations.

Table 4: Ablation study on the closed-loop benchmark.

Budget	Scene	Speed	Path	DS↑	SR(%)↑	Efficiency↑	Comfortness↑	Avg. B (%)↓
✓	✓	✓	✓	**84.87**	**65.45**	**266.19**	<u>54.36</u>	**44.64**
×	✓	✓	✓	83.81	62.55	<u>259.18</u>	51.49	60.03
✓	✓	×	✓	<u>84.60</u>	<u>64.09</u>	249.68	37.63	71.89
✓	✓	✓	×	83.88	63.63	247.17	30.75	66.70
✓	✓	×	×	84.12	63.18	241.10	34.24	68.05
✓	×	✓	✓	83.37	<u>64.09</u>	258.74	**56.74**	<u>54.37</u>
✓	×	×	×	84.04	63.18	254.06	35.35	73.59

Budget-conditioned Layer Routing. To demonstrate the superiority of our budget-conditioned mechanism, we compare our scene-adaptive allocation against models utilizing random budgets and

fixed budgets. As shown in Tab. 4, random allocation activates 60% of parameters on average yet yields performance inferior to our dynamic 45% activation. Furthermore, with an average activation of merely 45%, our method achieves a Success Rate falling between the 70% and 100% fixed-activation baselines as shown in Tab. 2. This shows that our scene-adaptive approach effectively reduces computational overhead in routine scenes while preserving essential capacity for complex scenarios.

4.5 Generalization and Safety-Critical Analysis

Generalization. Our primary experiments apply the Scene-Adaptive framework to SimLingo due to its strong closed-loop performance and efficiency. To evaluate generalization across VLA backbones, we further integrate SA into ORION, yielding ORION+SA. Unlike SimLingo, ORION adopts a Vicuna-v1.5 7B language backbone and a multi-view perception architecture.

As shown in Tab. 1 and Tab. 2, ORION+SA reduces TFLOPs by 47.4% and inference latency by 45.4%, while maintaining competitive driving performance and outperforming DriveMoE. Its remaining gap from full ORION mainly stems from route deviations rather than safety-critical failures, as discussed in the next section. Moreover, with only about 53% average activation, ORION+SA achieves DS and SR between those of the 65% and 100% fixed-activation baselines. These results demonstrate that SA generalizes to a substantially different VLA backbone, reducing computation in routine scenes while preserving sufficient capacity for complex ones.

Safety-Critical Analysis. Safety is critical for autonomous driving. To examine whether SA affects the safety-related capabilities of the original models, we analyze their failed Bench2Drive routes, focusing particularly on collision-related failures. We group failures into four categories and report the number of affected routes. Because a route may contain multiple violations, the category totals can exceed the total number of failed routes. Route Compliance (RC) mainly reflects route deviations, whereas Stability and Progress (S&P) and Traffic Rule (TR) capture route completion and traffic-rule compliance.

Table 5: Failed routes affected by each violation category.

Category	SimLingo	+SA	ORION	+SA
Collision	55.00	52.00	79.00	81.00
RC	13.00	19.00	26.00	37.00
S & P	17.00	13.00	12.00	10.00
TR	10.00	11.00	15.00	18.00

As shown in Tab. 5, collision-related failures do not increase noticeably for either SA model and even decrease for SimLingo+SA, indicating that SA does not introduce a substantial increase in observed collision-related failures. The main increase occurs in Route Compliance, which primarily reflects route-following stability rather than direct safety failures; the other categories remain comparable or improve slightly. Thus, despite the small decrease in overall driving performance, SA largely preserves the safety-critical driving capabilities of the original models.

5 Conclusions

This study presents Scene-Adaptive VLA, an efficient framework designed to mitigate the substantial computational overhead and energy consumption of VLA models in autonomous driving. Addressing the underexplored area of structural adaptation, the proposed methodology implements a dynamic budget-conditioned layer routing mechanism that conditionally activates model parameters based on the real-time scene. By quantifying frame-level temporal scene variations and decision deviations, the framework intelligently allocates computational budgets and selects optimal layer combinations through an architecture optimized end-to-end. Evaluations on the Bench2Drive benchmark demonstrate that when adapted to a SOTA VLA driving model, Scene-Adaptive VLA reduces average inference-time computational overhead by 53.5% with negligible degradation in driving and language capabilities.

Limitations and Future Work: Although our framework is theoretically applicable to any dense VLA driving model, broader validation was limited by available computational resources. Future work will evaluate SA on more diverse VLA backbones, incorporate target-route deviation and auxiliary route-following loss to reduce the performance gap, and explore joint sequential and structural adaptation for further efficiency improvements.

6 Acknowledgement

This work was supported in part by the National Key Research and Development Program of China under Grant 2023YFB4301900, in part by the Natural Science Foundation of China under Grant 62272053, Grant 62671071, and Grant 62472048, in part by the Beijing Nova Program under Grant 202604841213, and in part by the International Academic Exchange Program for Doctoral Students of Beijing University of Posts and Telecommunications.

References

[1] Zhenhua Xu, Yan Bai, Yujia Zhang, Zhuoling Li, Fei Xia, Kwan-Yee K Wong, Jianqiang Wang, and Hengshuang Zhao. Drivegpt4-v2: Harnessing large language model capabilities for enhanced closed-loop autonomous driving. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 17261–17270, 2025.

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

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

[4] Haoyu Fu, Diankun Zhang, Zongchuang Zhao, Jianfeng Cui, Dingkang Liang, Chong Zhang, Dingyuan Zhang, Hongwei Xie, Bing Wang, and Xiang Bai. Orion: A holistic end-to-end autonomous driving framework by vision-language instructed action generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 24823–24834, 2025.

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

[6] Zhenjie Yang, Yilin Chai, Xiaosong Jia, Qifeng Li, Yuqian Shao, Xuekai Zhu, Haisheng Su, and Junchi Yan. Drivemoe: Mixture-of-experts for vision-language-action model in end-to-end autonomous driving. *arXiv preprint arXiv:2505.16278*, 2025.

[7] Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22857–22867, 2025.

[8] Xuyang Liu, Ziming Wang, Junjie Chen, Yuhang Han, Yingyao Wang, Jiale Yuan, Jun Song, Siteng Huang, and Honggang Chen. Global compression commander: Plug-and-play inference acceleration for high-resolution large vision-language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 7350–7358, 2026.

[9] Xiaoyun Zhang, Jingqing Ruan, Xing Ma, Yawen Zhu, Haodong Zhao, Hao Li, Jiansong Chen, Ke Zeng, and Xunliang Cai. When to continue thinking: Adaptive thinking mode switching for efficient reasoning. *arXiv preprint arXiv:2505.15400*, 2025.

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

[11] Zhuoyan Xu, Khoi Duc Nguyen, Preeti Mukherjee, Saurabh Bagchi, Somali Chaterji, Yingyu Liang, and Yin Li. Learning to inference adaptively for multimodal large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3552–3563, 2025.

[12] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019.

[13] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[14] Anh-Quan Cao, Angela Dai, and Raoul De Charette. Pasco: Urban 3d panoptic scene completion with uncertainty awareness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14554–14564, 2024.

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

[16] Bo Jiang, Shaoyu Chen, Qing Xu, Bencheng Liao, Jiajie Chen, Helong Zhou, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. Vad: Vectorized scene representation for efficient autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8340–8350, 2023.

[17] SungYeon Park, MinJae Lee, JiHyuk Kang, Hahyeon Choi, Yoonah Park, Juhwan Cho, Adam Lee, and DongKyu Kim. Vlaad: Vision and language assistant for autonomous driving. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 980–987, 2024.

[18] Chenbin Pan, Burhaneddin Yaman, Tommaso Nesti, Abhirup Mallik, Alessandro G Allievi, Senem Velipasalar, and Liu Ren. Vlp: Vision language planning for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14760–14769, 2024.

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

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

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

[22] Chen Zhu, Wangbo Zhao, Huiwen Zhang, Yuhao Zhou, Weidong Tang, Shuo Wang, Zhihang Yuan, Yuzhang Shang, Xiaojiang Peng, Kai Wang, et al. Ea-vit: Efficient adaptation for elastic vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1038–1047, 2025.

[23] Yue Meng, Chung-Ching Lin, Rameswar Panda, Prasanna Sattigeri, Leonid Karlinsky, Aude Oliva, Kate Saenko, and Rogerio Feris. Ar-net: Adaptive frame resolution for efficient action recognition. In *European conference on computer vision*, pages 86–104. Springer, 2020.

[24] Daniel Rotem, Michael Hassid, Jonathan Mamou, and Roy Schwartz. Finding the sweet spot: Analysis and improvement of adaptive inference in low resource settings. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14836–14851, 2023.

[25] Zhangwei Gao, Zhe Chen, Erfei Cui, Yiming Ren, Weiyun Wang, Jinguo Zhu, Hao Tian, Shenglong Ye, Junjun He, Xizhou Zhu, et al. Mini-internvl: a flexible-transfer pocket multi-modal model with 5% parameters and 90% performance. *Visual Intelligence*, 2(1):32, 2024.

[26] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. arxiv 2024. *arXiv preprint arXiv:2407.10671*, 2024.

[27] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

[28] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[29] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

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

[31] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Spice: Semantic propositional image caption evaluation. In *European conference on computer vision*, pages 382–398. Springer, 2016.

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

A Details of Computational Overhead Evaluation

To specifically isolate and evaluate the impact of structural adaptation on inference computational overhead, we establish a fair and standardized configuration for input N_{in} and output N_{out} tokens across all evaluated models. Input Tokens (N_{in}): The input sequence comprises frame-level visual

Table 6: Token numbers across different models.

	Visual Tokens	Prompt Tokens	Specific Tokens
ORION	512	10	16
SimLingo	512	25	32
AutoVLA	1536	200	0
DriveMoE	512	10	5

tokens, language prompt tokens, and model-specific auxiliary tokens, with precise quantities inferred from their respective original papers, as shown in Tab. 6. The predominant factor influencing N_{in} is the visual token count, which is directly determined by the number of frames input per decision. For instance, AutoVLA processes four consecutive frames per decision, inherently resulting in a significantly larger input token sequence compared to other models that typically rely on a two-frame average. Output Tokens (N_{out}): it denotes the number of autoregressively generated output tokens, including language and action tokens where applicable. For models that predict actions non-autoregressively, N_{out} is set to 1, corresponding to a single action prediction step. For autoregressive models, we use the model-specific average number of output tokens per step on the Bench2Drive benchmark, thereby accounting for variations in output length across models.

To ensure a fair comparison for models possessing dynamic inference capabilities, we evaluate them under their minimal configurations. AutoVLA is evaluated in fast-thinking mode, bypassing textual reasoning and autoregressively generating solely action tokens. Similarly, DriveMoE dynamically selects input views, allocating two consecutive frames per view. For our evaluation, we conservatively configure DriveMoE to utilize a single-view input.

B Bench2Drive Multi-Ability Results

Multi-ability testing utilizes short, scenario-specific routes to isolate individual driving skills, effectively reducing metric variance and revealing specific system strengths. It assesses performance

Table 7: Multi-ability evaluation on Bench2Drive.

Method	**Ability (%) ↑**					**Mean**
	Merging	Overtaking	Emergency Brake	Give Way	Traffic Sign	
UniAD-Base	14.10	17.78	21.67	10.00	14.21	15.55
VAD	8.11	24.44	18.64	20.00	19.15	18.07
TCP-traj	8.89	24.29	51.67	40.00	46.28	34.22
Orion	25.00	**71.11**	78.33	30.00	69.15	54.72
DriveMoE	34.67	40.00	65.45	40.00	59.44	47.91
simlingo	**54.01**	57.04	**88.33**	**53.33**	**82.45**	**67.03**
SimLing+SA	<u>47.50</u>	<u>68.88</u>	<u>80.00</u>	<u>50.00</u>	<u>80.52</u>	<u>65.38</u>

across distinct interactive scenarios, including merging, overtaking, Emergency Brake, Give Way, and Traffic Sign, and provides a comprehensive and decoupled evaluation of a model's driving capabilities under varied conditions. Although our method decreases the multi-ability mean score of the original model by 1.65%, it significantly enhances its overtaking capability by 11.84%, as shown in Tab. 7.

C Training Details

We present the pseudocode for the two-stage training pipeline. The notation follows Section 3.

C.1 Stage 1: Scene-Aware Representation and Layer Routing Initialization

Stage 1 jointly initializes the scene-aware encoders, the routing scheduler, the budget encoder, the layer routing query, and the LoRA-tuned prefix layers via supervised fine-tuning with randomized budgets. The budget scheduler f_{budget} and budget query $\mathbf{q}_b$ remain frozen. Algorithm 1 details the procedure, with the Straight-Through Gumbel-Softmax top-k estimator highlighted.

Algorithm 1 Stage 1: Scene-Aware Representation and Layer Routing Initialization

Input: Training dataset $\mathcal{D} = \{(\mathbf{e}_I, p_{\text{task}}, \mathbf{e}_{\text{nav}}, v, \mathbf{W}_p^{\text{gt}}, \mathbf{W}_s^{\text{gt}}, l_{1:T})\}$, Prefix layers L_{pre}, Sub-layers L_{sub}, Budget lower bound $b_{\min}$, Temperature τ, Loss weights $\lambda_{\text{wp}}, \lambda_{\text{lang}}, \lambda_{\text{div}}, \lambda_{\text{bal}}$

Output: Optimized parameters for scene-adaptive routing Θ_{train}

Initialize trainable parameters $\Theta_{\text{train}} \leftarrow \{\mathbf{q}_{lr}, f_{\text{routing}}, f_{\text{enc}}, \text{scene encoders}, \text{prefix LoRA}\}$
Freeze budget scheduler and query $\Theta_{\text{frozen}} \leftarrow \{f_{\text{budget}}, \mathbf{q}_b\}$
for each training step **do**
  Sample batch from $\mathcal{D}$
  Extract scene-aware tokens $\mathbf{e}_s, \mathbf{e}_{pw}, \mathbf{e}_{sw}$ via Eq. (1)–(4)
  Form input sequence $I_{\text{llm}} \leftarrow [p_{\text{global}}, \mathbf{q}_p, \mathbf{q}_s, \mathbf{e}_s, \mathbf{e}_{pw}, \mathbf{e}_{sw}, \mathbf{q}_b, \mathbf{q}_{lr}]$
  Extract budget feature $\mathbf{h}_{q_b} \leftarrow \text{PrefixLayers}_{1 \to L_{\text{pre}}/2}(I_{\text{llm}})$
  Sample random budget $b \sim \mathcal{U}[b_{\min}, 1]$
  Quantize active layer count $k \leftarrow \max(0, \lfloor (L_{\text{pre}} + L_{\text{sub}}) \cdot b \rfloor - L_{\text{pre}})$
  Encode budget $\mathbf{e}_b \leftarrow f_{\text{enc}}(\text{Sinusoidal}(b))$ and replace $\mathbf{q}_b$ in intermediate features
  Updated features $\mathbf{h}_{q_{lr}} \leftarrow \text{PrefixLayers}_{L_{\text{pre}}/2+1 \to L_{\text{pre}}}$
  Compute routing logits $\boldsymbol{\pi}_{\text{layer}} \leftarrow f_{\text{routing}}([\mathbf{h}_{q_{lr}}, \mathbf{e}_b]) \in \mathbb{R}^{L_{\text{sub}}}$
  Straight-Through Gumbel-Softmax Top-k estimation:
    $\mathbf{s} \leftarrow \text{softmax}\big((\boldsymbol{\pi}_{\text{layer}} + \mathbf{g})/\tau\big), \mathbf{g} \sim \text{Gumbel}(0, 1)$
    $\mathbf{M}_{\text{hard}} \leftarrow \text{OneHotTopK}(\mathbf{s})$
    Mask $\mathbf{M} \leftarrow \texttt{stopgrad}(\mathbf{M}_{\text{hard}} - \mathbf{s}) + \mathbf{s}$
  Route and decode $\mathbf{W}_p, \mathbf{W}_s, l_{1:T} \leftarrow \text{SubLayers}(\text{features} \mid \mathbf{M})$
  Compute $\mathcal{L}_{\text{wp}}, \mathcal{L}_{\text{lang}}$ via Eq. (6) and $\mathcal{L}_{\text{div}}, \mathcal{L}_{\text{bal}}$ via Eq. (7)
  $\mathcal{L}_{\text{stage1}} \leftarrow \lambda_{\text{wp}}\mathcal{L}_{\text{wp}} + \lambda_{\text{lang}}\mathcal{L}_{\text{lang}} + \lambda_{\text{div}}\mathcal{L}_{\text{div}} + \lambda_{\text{bal}}\mathcal{L}_{\text{bal}}$
  Optimize Θ_{train} via $\nabla_{\Theta_{\text{train}}}\mathcal{L}_{\text{stage1}}$
end for

The diversity regularizer uses max-pooled scene features $\mathbf{z}_i$ aggregated from visual tokens, prompt tokens, and the budget token: $\mathbf{z}_i = \max\text{pool}\big(\{\mathbf{e}_I^{i,j}\}_{j=1}^{N_c} \cup \{\mathbf{e}_{\text{prompt}}^{i,t}\}_t \cup \{\mathbf{e}_b^i\}\big)$. The cosine similarity between routing masks $\mathbf{M}_i$ is contrasted with that of scene features $\mathbf{z}_i$: $\text{sim}(\mathbf{M}_i, \mathbf{M}_j) = \frac{\mathbf{M}_i \cdot \mathbf{M}_j}{\|\mathbf{M}_i\|_2 \|\mathbf{M}_j\|_2}$, and analogously for $\text{sim}(\mathbf{z}_i, \mathbf{z}_j)$.

C.2 Stage 2: GRPO with Budget Allocation Policy

Stage 2 freezes the routing pathway and optimizes the budget allocation policy π_θ via Group Relative Policy Optimization (GRPO). Crucially, since the budget query feature $\mathbf{h}_{q_b}$ is extracted midway through the prefix layers, we explicitly unfreeze only the first half of the prefix layers (1 to $L_{\text{pre}}/2$) alongside the budget scheduler, while freezing the latter half to preserve the routing stability established in Stage 1. The policy $\pi_\theta(a \mid \mathbf{h}_{q_b}) = \text{Categorical}\big(\text{softmax}(f_{\text{budget}}(\mathbf{h}_{q_b}))\big)$ maps the intermediate budget query representation to a distribution over K discrete budget actions. Algorithm 2 presents the procedure.

Algorithm 2 Stage 2: Reinforcement Fine-Tuning for Budget Allocation

Input: Training dataset $\mathcal{D}$, Initial policy π_θ from Stage 1, Group size G, Update epochs E, Budget mapping $b(a)$, Reward weights w_d, w_b, w_a, Clipping range ε_c, KL weight β, Anchor weight β_{anchor}
Output: Optimized budget allocation policy π_θ

Unfreeze policy parameters $\theta \leftarrow \{f_{\text{budget}}, \mathbf{q}_b, \text{scene encoders, prefix LoRA}_{1 \to L_{\text{pre}}/2}\}$
Freeze routing parameters $\Theta_{\text{frozen}} \leftarrow \{f_{\text{routing}}, f_{\text{enc}}, \mathbf{q}_{lr}, \text{base LLM, prefix LoRA}_{L_{\text{pre}}/2+1 \to L_{\text{pre}}}\}$
Store reference anchor parameters $\bar{\boldsymbol{\phi}} \leftarrow \boldsymbol{\phi}_{\text{Stage1}}$ and initial policy $\pi_{\text{old}} \leftarrow \pi_\theta$
for each training step **do**
  Sample batch from $\mathcal{D}$ and extract budget query features $\{\mathbf{h}_{q_b}^{(i)}\}_{i=1}^{B}$
  for each sample $i \in \{1, \ldots, B\}$ **do**
    Sample G actions $\{a_{i,g}\}_{g=1}^{G} \sim \text{Categorical}\big(\text{softmax}(f_{\text{budget}}(\mathbf{h}_{q_b}^{(i)}))\big)$ under π_{old}
    Execute forward pass for each $a_{i,g}$ to obtain driving loss $\mathcal{L}_{i,g}$
    Compute reward $r_{i,g}$ and relative advantage $A_{i,g}$ via Eq. (8)
  end for
  for epoch $= 1$ to E **do**
    Evaluate importance sampling ratio $\rho_{i,g}(\theta) \leftarrow \frac{\pi_\theta(a_{i,g}|\mathbf{h}_{q_b}^{(i)})}{\pi_{\text{old}}(a_{i,g}|\mathbf{h}_{q_b}^{(i)})}$
    Compute policy objective $\mathcal{L}_{\text{policy}}$ via Eq. (9)
    Compute anchor loss $\mathcal{L}_{\text{anchor}} \leftarrow \frac{1}{|\boldsymbol{\phi}|}\sum_j (\phi_j - \bar{\phi}_j)^2$
    Evaluate task driving loss $\mathcal{L}_{\text{driving}} \leftarrow \lambda_{\text{wp}}\mathcal{L}_{\text{wp}} + \lambda_{\text{lang}}\mathcal{L}_{\text{lang}}$ under current π_θ
    $\mathcal{L}_{\text{stage2}} \leftarrow \mathcal{L}_{\text{policy}} + \beta_{\text{anchor}}\mathcal{L}_{\text{anchor}} + w_{\text{driving}}\mathcal{L}_{\text{driving}}$
    Optimize θ via $\nabla_\theta \mathcal{L}_{\text{stage2}}$
  end for
  Synchronize reference policy $\pi_{\text{old}} \leftarrow \pi_\theta$
end for

The direct driving loss weight follows a linear warmup, preventing early-stage policy collapse. The anchor parameters $\bar{\boldsymbol{\phi}}$ are snapshots of the corresponding Stage 1 trained weights and are never updated during Stage 2. The budget action space consists of $K = 10$ discrete actions, where action $a \in \{0, \ldots, 9\}$ maps to budget $b(a) = \max(0.1 \cdot (a+1),\, b_{\min})$. To encourage exploration in early Stage 2, we optionally apply temperature annealing and ϵ-greedy sampling when drawing actions from the old policy.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction are empirically verified in Section 4, in particular Table 2.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Our limitations are discussed in Section 5.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We disclose all information needed for reproduction in Section 4 and Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will provide data, code, and instructions for the final paper.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide experimental details in Section 4 and Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We conduct our experiments with 3 random seeds and report the mean of the results as detailed in Section 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We illustrate the computer resources we use in Section 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We conducted the research in the paper in conformance with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Our work contributes to the advancement of VLA-based autonomous driving technology as introduced in Section 1.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [No]

    Justification: Our paper does not release data or models that have a high risk for misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use publicly available datasets and models with proper citations and respect for their licenses.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We will release our codebase along with detailed README files for our work.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as an important, original, or non-standard component.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.