Can Small Quantized VLMs Drive? An Experimental Evaluation of Small Quantized VLMs for Autonomous Driving

Samson Mathew
Undergraduate Student
Hofstra University, USA
smathew5@pride.hofstra.edu

Zonghua Gu
Hofstra University, USA
zonghua.gu@hofstra.edu

***Abstract*—Recent advancements in Foundation Models (e.g., Vision Language Models, VLMs) have sparked strong interest in their applications in Autonomous Driving. However, their practical adoption is hindered by challenges in deployment and real-time inference of large VLMs on in-vehicle computing platforms with limited hardware resources, as they are quite resource-demanding due to their sheer size. DriveBench is a recently developed large-scale benchmark dataset, designed to evaluate VLM reliability for Autonomous Driving. In this study, we evaluate a number of small VLMs (either with or without weight quantization) on the dataset, and attempt to answer the research question *Can small and quantized VLMs drive?* Although the general conclusion may be negative, this work aims to provide a useful reference for researchers and practitioners for evaluating the trade-offs when selecting an appropriate VLM for Autonomous Driving.**

***Index Terms*—Foundation Models, Vision Language Models, Model Quantization, Autonomous Driving**

I. Introduction

Foundation Models (FMs) are Deep Learning models that are pre-trained on large unlabeled datasets through self-supervision and then fine-tuned for different downstream tasks, including Large Language Models (LLMs), Vision Language Models (VLMs), Diffusion Models, etc. FMs bring generalist knowledge gained during pre-training to enable human-level intelligence. VLMs are multimodal FMs that take image and text inputs, and generate image or text outputs, with a wide range of applications such as Question Answering, Sentiment Analysis, Image Captioning, Object Recognition, Semantic Segmentation, and Image/Video Generation. Recently, researchers have been exploring the application of VLMs to Autonomous Driving (AD). VLMs promise unified perception, reasoning, and decision support by combining visual understanding with natural language. Their reasoning and interpretive capabilities allow them to work at a high level for richer scene understanding and decision support to improve safety and efficiency across a wide range of driving scenarios, especially the corner cases where training data are sparse. This makes VLMs appealing for AD tasks spanning perception, prediction, planning, and explanation [1], [2].

A key challenge of practical deployment of large VLMs is their extremely high demand for compute and memory resources. Model size may range from large-scale models with over a trillion parameters, e.g., GPT-4, to small-scale models with a few billion parameters. For example, serving a 175 billion model requires at least 350 GB of GPU memory. (The largest single GPU memory currently is 80 GB (NVIDIA A100). Multi-GPU systems can aggregate to hundreds of GBs through NVLink.) The sheer size of large VLMs poses significant hurdles to their local deployment and efficient inference on resource-constrained in-vehicle hardware platforms.

Fig. 1 illustrates deployment of VLMs for AD in cloud vs. in vehicle. Large VLMs can be deployed in the cloud, and only small-scale VLMs can be deployed in the vehicle. Cloud deployment has fast inference speed, but may experience long and unpredictable wireless network latency, even with today's 5G and 6G wireless technologies. In-vehicle deployment suffers from slow inference speed due to limited hardware resources such as GPU availability, processor speed, memory size, etc. Cloud deployment may be practical for certain non-time-critical tasks, but in-vehicle deployment is preferred for achieving real-time inference at high frame rates in safety-critical systems such as AVs. As closed-loop control

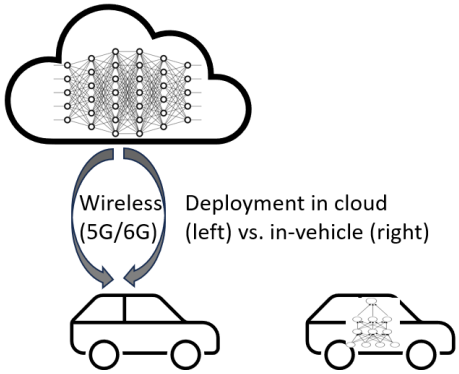


Fig. 1: Cloud deployment vs. local in-vehicle deployment.

This work was supported by the Advanced Summer Program in Research (ASPiRe) program at Hofstra University.

systems in an AV have stringent real-time constraints to ensure safety, VLMs, whether in-cloud or in-vehicle, are not ready to be used to replace traditional closed-loop controllers. Instead, they are typically used as a slow system that runs in parallel with the fast system that runs the perception-planning-control loop at a fast rate (say 30 Hz), forming a dual-system architecture. Inspired by Kahneman's dual-process theory, the fast system provides rapid, reactive outputs, and the slow system conducts deeper reasoning, contextual analysis, and complex or novel situation handling, typically with higher computational cost and slower throughput. These systems may operate synchronously or asynchronously, sharing representations and informing each other's outputs. While this architecture helps to make the timing requirements (deadlines) less stringent for VLM inference, it is still necessary to let the VLMs run periodically and in real-time.

Since only small and efficient VLMs can be practically deployed in-vehicle, we pose the research question: *Can small and quantized VLMs drive?*, i.e., can small-scale, weight-quantized VLMs that can be practically deployed on an in-vehicle hardware platform perform well despite their small size and high efficiency? In this paper, we consider eight small-scale VLMs and their 4-bit quantized versions, and measure their performance on an extensive AD dataset, in order to answer the above research question.

This paper is structured as follows: We introduce background and related work in Section II; report our methods in Section III; present the experimental results in Section IV; and conclusions and future work in Section V.

II. Background and Related Work

A. VLMs for AD

Application scenarios of VLM for AD include perception and scene understanding [3], trajectory prediction, planning and decision-making [4], human–vehicle interaction [5], policy adaptation and rule compliance [6], etc. General-purpose VLMs can be used directly "out-of-the-box" for AD tasks, or they can be used after fine-tuning with datasets in the target application domain. For maximum performance and efficiency, many specialist VLMs for AD [1] have been developed. (We limit our attention to general-purpose VLMs in this paper, and leave the evaluation of specialist VLMs as part of future work.)

There are a number of datasets/benchmarks for AD. Two notable recent datasets for VLM evaluation are Driving with language (DriveLM) [3] and DriveBench [7]. DriveLM proposes graph VQA as a proxy for human-like reasoning with a VLM baseline that supports end-to-end policy learning, by instantiating graph-structured QA spanning perception, prediction, and planning built atop the nuScenes dataset [8] and the CARLA driving simulator. DriveBench [7] builds upon DriveLM-nuScenes, and provides a reliability- and grounding-oriented benchmark across 17 conditions (clean, corrupted, and "no image"), revealing gaps in multimodal reasoning and robustness of VLMs for safety-critical systems. VLMs are shown to produce plausible yet weakly grounded answers, especially under degraded or missing visual inputs, underscoring the risks in applying VLM to AD domain. We use the DriveBench dataset for testing and performance evaluation in this paper (and plan to use the DriveLM dataset for model fine-tuning as part of future work).

B. Model Quantization

Model compression techniques aim to reduce the size, computation, and memory footprint of deep learning models while minimizing performance degradation. Well-known model compression techniques for LLMs/VLMs [9] include quantization, pruning, knowledge distillation, low-rank factorization, Neural Architecture Search, sparse models/Mixture-of-Experts. Model quantization is the process of reducing the precision of weights and/or activations to improve efficiency in terms of memory and inference speed. Model quantization techniques can be categorized into Post-Training Quantization (PTQ), e.g., Activation-aware Weight Quantization (AWQ) [10] and SmoothQuant [11], vs. Quantization-Aware Training (QAT), e.g., EfficientQAT [12].

In this paper, we use the BitsAndBytes library [13], which provides efficient 8-bit and 4-bit PTQ of model weights through on-the-fly dequantization. During inference, model weights are loaded in 4-bit precision into GPU memory and dynamically dequantized to FP16 or FP32 before matrix multiplications. This method considerably reduces memory footprint and enables larger models or batch sizes to be deployed on constrained hardware, while maintaining numerical precision and stability during computation. However, because dequantization occurs at runtime, the arithmetic operations still execute at higher precision, and any efficiency gains from reduced memory bandwidth can be offset by the added dequantization overhead and non-optimized memory access patterns, hence overall performance improvements are not guaranteed.

III. Method

A. Selected Models

We select 9 VLMs for our experiments as shown in Table I, including SmolVLM, Qwen2.5-VL, Gemma-3, and LLaVA-1.5 families. The table includes the model name on Huggingface and number of parameters for each VLM (the number of parameters is approximate but not far from the accurate number of parameters). The models are selected based on the following criteria: open source; small size (as there is no universal definition of small VLMs, we select models with 13 billion parameters or less); well-known models from major companies, downloadable from Huggingface. For each model, we always select the instruction-tuned version. (Instruction tuning is a supervised learning technique used to refine models by training them on labeled datasets that pair instructional prompts with desired outputs.) For each model, we also consider the 4-bit quantized models, named by appending "-4bit" to the corresponding model name, using on-the-fly weight quantization through the BitsAndBytes library [13], so a total of 18 VLMs are evaluated.

TABLE I: VLMs considered in this paper.

Model name in this paper	Model name on Huggingface	Params
SmolVLM-256M [14]	SmolVLM-256M-Instruct	256 M
SmolVLM-500M [14]	SmolVLM-500M-Instruct	500 M
SmolVLM-2.2B [14]	SmolVLM-Instruct	2.2 B
Qwen2.5VL-3B [15]	Qwen2.5-VL-3B-Instruct	3 B
Qwen2.5VL-7B [15]	Qwen2.5-VL-7B-Instruct	7 B
Gemma3-4B [16]	gemma-3-4b-it	4 B
Gemma3-12B [16]	gemma-3-12b-it	12 B
Llava1.5-7B [17]	llava-1.5-7b-hf	7 B
Llava1.5-13B [17]	llava-1.5-13b-hf	13 B

B. Hardware Platform and Software Framework

All experiments are performed on the Star cluster (https://starhpc.hofstra.io/). a High-Performance Computing system at Hofstra University. Each compute node contains an AMD EPYC 7513 Processor, and 8 SXM NVIDIA A100 GPUs with 1024 GiB total memory (16 x 64GiB DIMM DDR4). We use the vLLM library for LLM inference and serving [18], as used in DriveBench. We use offline inference, which means invoking the model directly from Python via its API with batched requests and predictions (as opposed to online inference, where the model is run as an API server, allowing clients to send requests over the network.) We test each model in both full-precision (FP16) and 4-bit quantized configurations to examine the trade-offs between GPT score and computational latency. For each model, we record fine-grained latency metrics, including prefill, decode, and end-to-end inference times, averaged across multiple question types and datasets to capture overall runtime behavior. Prefill time corresponds to the forward-pass computation required to process the input prompt and populate the key–value (KV) cache before any tokens are generated, while decode time measures the total latency spent generating output tokens sequentially. This distinction allows for an accurate breakdown of parallelizable versus sequential components of inference latency.

C. Performance Metrics

We adopt the DriveBench [7] dataset for vision-based AD, where each input consists of images from 6 cameras surrounding the vehicle: Front left, Front, Front Right, Back left, Back, Back Right. Similar to DriveBench [7] and DriveLM [3], we mainly consider the GPT score as the metric for evaluating the quality of the VLM output[1]. GPT Score is an LLM-as-a-judge evaluation approach that uses an LLM to assign a quantitative score (0-100) to generated text. We adopt OpenAI GPT-5-mini for GPT Score evaluation. GPT-5-mini is a compact version of GPT-5, released in August 2025, designed to handle lighter-weight reasoning tasks[2]. The model prompt consists of detailed rubrics that account for answer correctness, coherence, and alignment of explanations with the final answer, and we omit the details here which are described in [7].

Based on each original clean image, DriveBench provides 15 different corruption types, including weather conditions (Brightness, Dark, Fog, Snow, and Rain), external disturbances (Water Splash and Lens Obstacle), sensor failures (Camera Crash, Frame Lost, and Saturate), motion blurs (Motion Blur and Zoom Blur), and data transmission errors (Bit Error, Color Quant, and H.265 Compression). In addition, they include a black "noimage" input, i.e., when the visual information is absent and the car is essentially "driving blind". They observed that some models performed well even for the "noimage" input, suggesting that the model response might be based on majority biases (e.g., Going Ahead, the most common action in most driving scenarios) instead of camera input. For each model, we compute its *Average GPT Score* for each of the 17 input types by averaging the GPT scores across all 1,261 questions spanning four driving tasks of perception, prediction, planning, and behavior. We further compute its *Overall Average GPT Score* by averaging across all 17 input types, as an overall performance metric for each model.

For timing performance, vLLM provides built-in performance metrics including: Server-level global metrics that track the state and performance of the LLM engine; and request-level metrics that track the characteristics (e.g. size and timing) of individual requests. We focus on request-level metrics, accessed with the function call `llm.get_metrics()` (https://docs.vllm.ai/en/stable/examples/offline_inference/metrics.html).

IV. Experimental Results

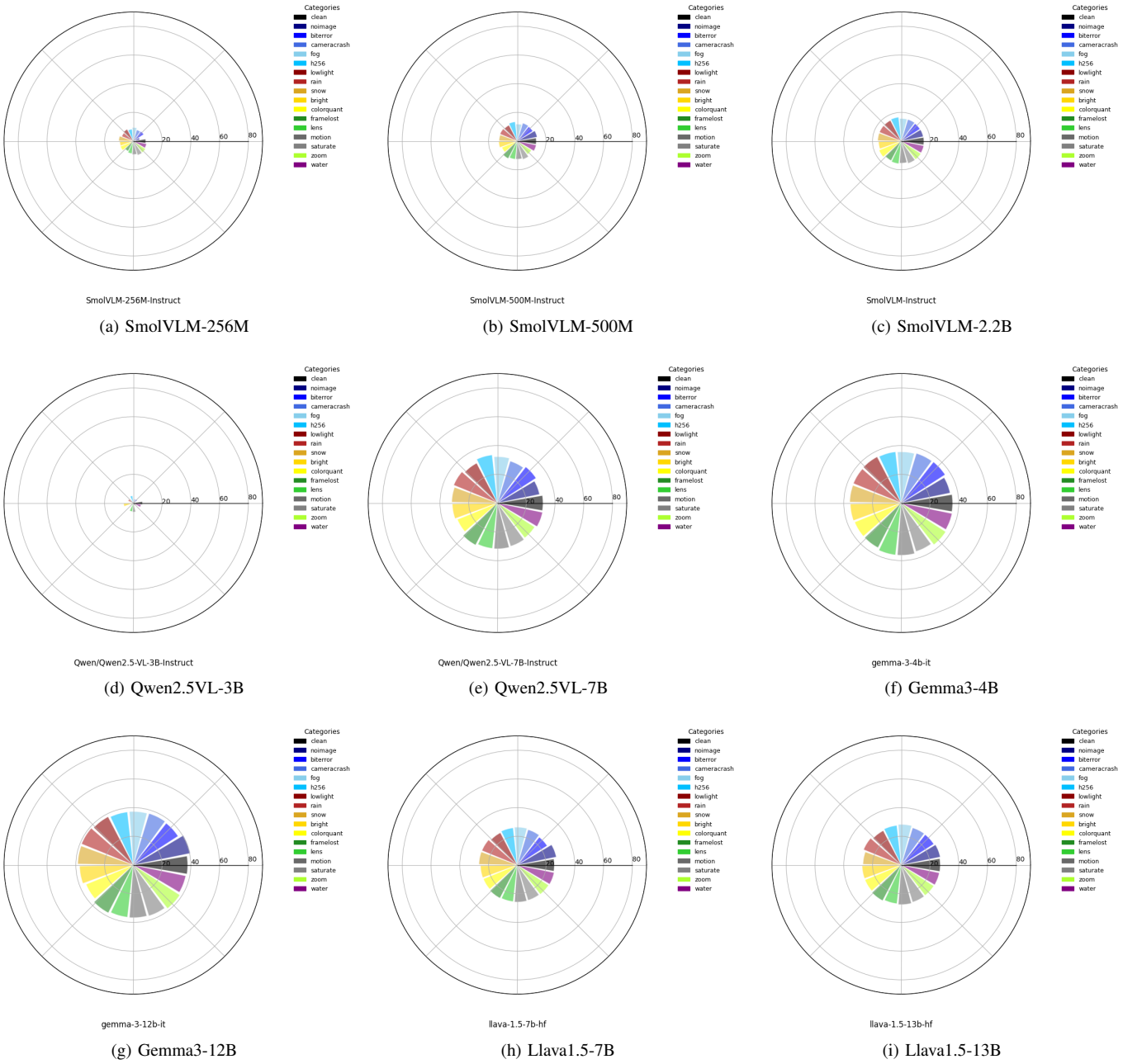
A. GPT Scores

We plot the Average GPT Score of each model for the 17 input types as radar graphs (or polar graphs). Fig. 2 shows the results for full-precision models, and Fig. 3 shows results for 4-bit quantized models. To facilitate high-level comparison among the different models, Fig. 5 plots the Overall Average GPT Scores, obtained by averaging the Average GPT Scores of each model across all 17 input types and all requests for each input type, i.e., by further averaging the scores across all 17 input types for each Radar graph in Figs. 2 and 3. We make the following observations from these figures:

- While larger models generally perform better, model performance is more correlated with model series (e.g., the Gemma-3 model series has the best overall performance) than with different model sizes with a series. (One exception is the Qwen2.5-VL series, where the 3B-Instruct model has significantly worse performance than the 7B-Instruct model.)

[1]While some traditional language metrics based on pattern-matching between predicted responses and ground-truth answers, such as BLEU [19] and ROUGE-L [20], are also used in [7], they are shown to poorly reflect underlying key information, and GPT scores are shown to be a better metric overall.

[2]We did not use the full version of GPT-5 for budget/cost reasons, since we found that GPT-5 and GPT-5-mini give similar results for selected samples, so GPT-5-mini is adequate for our purposes.



(a) SmolVLM-256M (b) SmolVLM-500M (c) SmolVLM-2.2B

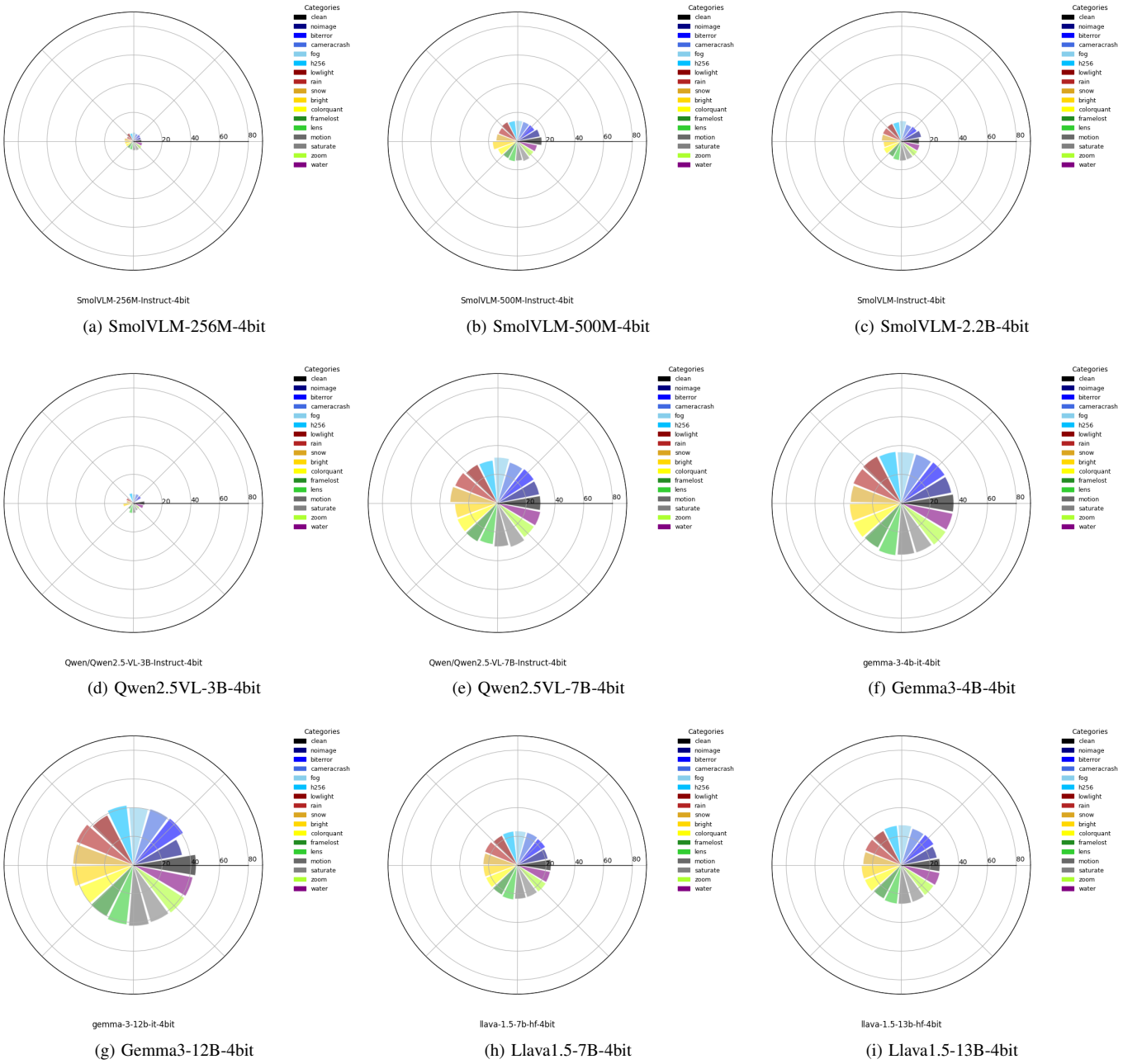
(d) Qwen2.5VL-3B (e) Qwen2.5VL-7B (f) Gemma3-4B

(g) Gemma3-12B (h) Llava1.5-7B (i) Llava1.5-13B

Fig. 2: Radar graph of Average GPT Score for each model with full-precision weights.

- For most models, model quantization has the effect of reducing performance, but the performance reduction is generally not very large, and some 4-bit models achieve slightly improved performance compared to their full-precision counterparts. We used a simple PTQ libary BitsAndBytes in this paper, and we expect the performance gap to be even smaller with more sophisticated quantization techniques.
- Among all 18 model variants, Gemma3-12B-4bit has the highest Overall Average GPT Score of 41.2, and Qwen2.5VL-3B has the lowest score of 3.1. Upon closer examination, we find that Qwen2.5VL-3B (and its 4-bit version) provides blank responses to many requests and inputs, thus receiving a score of 0 for them.
- Even for the highest-performing model, the Overall Average GPT Score of 41.2 is far below the bar for effective driving (which should be in the upper ninety percent range), hence none of the 18 models is good enough for practical AD tasks without further performance improvements (e.g., with model fine-tuning or custom model designs).

While there are some overlaps between the models in this paper and those included in the DriveBench paper [7], our results are not comparable with the radar graphs in [7] for



(a) SmolVLM-256M-4bit (b) SmolVLM-500M-4bit (c) SmolVLM-2.2B-4bit

(d) Qwen2.5VL-3B-4bit (e) Qwen2.5VL-7B-4bit (f) Gemma3-4B-4bit

(g) Gemma3-12B-4bit (h) Llava1.5-7B-4bit (i) Llava1.5-13B-4bit

Fig. 3: Radar graph of Average GPT Score for each model with 4-bit quantization.

two reasons: First, we use the latest version of ChatGPT (GPT-5-mini) for GPT Score evaluation, whereas [7] uses GPT-3.5-turbo. While the two versions of ChatGPT tend to give similar scores for most inputs, they sometimes give significantly different scores for the same input. (The reason for adopting GPT-5-mini is that we found that GPT-3.5-turbo frequently hangs during evaluation of some models with poor output quality, e.g, with repeated phrases as shown in the Appendix, but GPT-5-mini does not have this issue.) Second, the radar graphs in Figure 25 in [7] are actually the GPT scores for the Planning task, not the multi-task averages as stated in the caption, which should be derived from Table 2.

B. Timing Metrics

Fig. 5 plots the overall average request inference times, obtained by averaging the inference times of each model across all 17 input types and all requests for each input type[3]. It also

[3]vLLM also provides the end-to-end latency metric, which is equal to Inference Time + Queue Time + Misc Overhead. We found that Queue time is almost negligible, meaning batching and scheduling are efficient. Misc Overhead is small but not negligible, which accounts for orchestration costs, i.e., synchronization, minor I/O, or framework-level delays not counted inside GPU inference time.

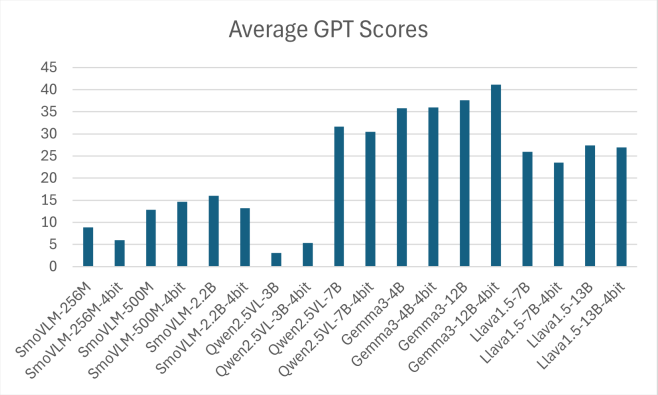


Fig. 4: Overall Average GPT Score for each model.

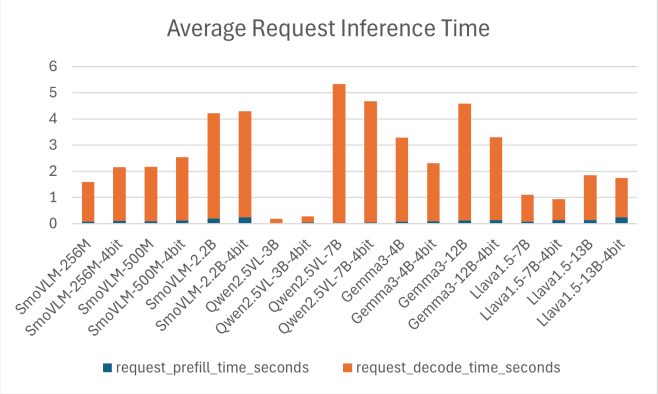


Fig. 5: Overall Average Inference Time for each model.

shows the breakdown of the inference time as the sum of prefill time (encode/process input and build KV cache) and decode time (autoregressive token generation). We make the following observations regarding the timing metrics:

- Across all model families, the decode phase dominates total inference latency, typically accounting for over 90% of the total request time. Most models have between decode-to-prefill ratio of 20× and 40×, with smaller or quantized models tending toward the lower end. This reflects the inherent sequential nature of token generation during decoding, whereas prefill (context processing) is largely parallelizable and scales more efficiently with model and hardware size. Hence decoding efficiency is the principal bottleneck for real-time LLM/VLM inference.
- Larger models (e.g., Gemma-12B and Qwen2.5-VL-7B) exhibit longer decode times, consistent with their increased parameter counts and attention complexity. Meanwhile, smaller models (e.g., SmolVLM-256M and LLaVA-1.5-7B) demonstrate much shorter inference times, making them more lightweight and suitable for deployment on resource-constrained platforms.
- Qwen2.5VL-3B and Qwen2.5VL-3B-4bit have very short average inference times, since they tend to produce very short responses (˜26 tokens each), while the larger 7B variants generate much longer outputs (˜511 tokens), with lengthy explanations in the output. This is consistent with the low Average GPT Scores of Qwen2.5VL-3B and Qwen2.5VL-3B-4bit in Fig. 4.
- Across question types (planning, perception, prediction, behavior), prefill times remain relatively stable for a given model, whereas decode times vary more since some question types trigger longer responses, since prefill time is function(prompt_length), whereas decode time is function(output_length). (We omit listing of detailed timing metrics for each question type for space limitations.)
- Quantized models (4-bit variants) exhibit mixed inference performance. Although on-the-fly quantization using BitsAndBytes [13] significantly reduces memory consumption and enables larger batch sizes or deployment of higher-capacity models on constrained hardware, it does not always lead to reduced latency, as discussed in Section II-B. In some cases, the 4-bit versions of smaller models even show slightly longer inference times than their full-precision counterparts, since smaller models are typically more compute-bound than memory-bound and thus benefit less from the reduced data-transfer overhead of on-the-fly quantization. These findings suggest that on-the-fly quantization primarily enhances memory and deployment efficiency, rather than guaranteeing faster runtime performance. To achieve greater end-to-end efficiency, offline quantization combined with hardware-level support for low-precision arithmetic is needed.

V. Conclusions and Future Work

In this paper, we evaluated a diverse set of small and quantized VLM models on a comprehensive dataset DriveBench, across clean and corrupted conditions for 4 common driving tasks. The answer is broadly negative at present: out-of-the-box small and quantized VLMs are generally not sufficiently reliable to be used for AD tasks safely, due to their small sizes and limited training datasets. Nevertheless, certain VLMs (e.g., Gemma-3 series) achieve good performance despite their small size, and are strong candidates for continued research to improve their performance and efficiency. Model quantization may result in a small reduction in performance, but the performance reduction is generally insignificant, and sometimes it may even help to improve performance. Given proper hardware support, model quantization can result in significant inference speedups, hence it is a promising technique for improving model efficiency with small performance degradation.

Open challenges and directions include:

- Domain alignment with fine-tuning: Model fine-tuning of VLMs on domain-specific datasets can help improve grounding and reduce hallucinations for that domain, e.g., driving tasks such as perception, prediction, and planning. We plan to consider Parameter Efficient Fine-Tuning (PEFT) techniques like LoRA (Low-Rank Adaptation) [21] for fine-tuning LLMs/VLMs without retraining all their weights.

- Specialist models vs. general-purpose models: while this paper focused on general-purpose VLMs, we plan to consider specialist VLMs in the future, which can achieve good performance with relatively small sizes [1].
- Efficient perception: Visual token pruning and salience-guided attention can reduce computing time while prioritizing safety-critical cues, e.g., foreground objects such as cars and pedestrians should be given higher priority than background objects such as trees, grass and the sky [22].
- Vision-Language-Action (VLA) [2]: VLA is the next-generation evolution beyond VLM in AD, extending VLM by coupling vision-language understanding with actionable driving control and enabling a more complete and end-to-end framework for AD.

ACKNOWLEDGMENTS

We would like to thank the authors of DriveBench [7] for their help and support in resolving some technical issues.

REFERENCES

[1] X. Zhou, M. Liu, E. Yurtsever, B. L. Zagar, W. Zimmer, H. Cao, and A. C. Knoll, "Vision language models in autonomous driving: A survey and outlook," *IEEE Transactions on Intelligent Vehicles*, 2024.

[2] S. Jiang, Z. Huang, K. Qian, Z. Luo, T. Zhu, Y. Zhong, Y. Tang, M. Kong, Y. Wang, S. Jiao *et al.*, "A survey on vision-language-action models for autonomous driving," *arXiv preprint arXiv:2506.24044*, 2025.

[3] C. Sima, K. Renz, K. Chitta, L. Chen, H. Zhang, C. Xie, J. Beißwenger, P. Luo, A. Geiger, and H. Li, "Drivelm: Driving with graph visual question answering," in *European conference on computer vision*. Springer, 2024, pp. 256–274. [Online]. Available: https://opendrivelab.com/DriveLM

[4] C. Pan, B. Yaman, T. Nesti, A. Mallik, A. G. Allievi, S. Velipasalar, and L. Ren, "Vlp: Vision language planning for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 14 760–14 769.

[5] T. Deruyttere, S. Vandenhende, D. Grujicic, L. Van Gool, and M.-F. Moens, "Talk2car: Taking control of your self-driving car," *arXiv preprint arXiv:1909.10838*, 2019.

[6] B. Li, Y. Wang, J. Mao, B. Ivanovic, S. Veer, K. Leung, and M. Pavone, "Driving everywhere with large language model policy adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 14 948–14 957.

[7] S. Xie, L. Kong, Y. Dong, C. Sima, W. Zhang, Q. A. Chen, Z. Liu, and L. Pan, "Are vlms ready for autonomous driving? an empirical study from the reliability, data, and metric perspectives," *arXiv preprint arXiv:2501.04003*, 2025. [Online]. Available: https://drive-bench.github.io

[8] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[9] X. Zhu, J. Li, Y. Liu, C. Ma, and W. Wang, "A survey on model compression for large language models," *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 1556–1577, 2024.

[10] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han, "Awq: Activation-aware weight quantization for on-device llm compression and acceleration," *Proceedings of machine learning and systems*, vol. 6, pp. 87–100, 2024.

[11] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "Smoothquant: Accurate and efficient post-training quantization for large language models," in *International conference on machine learning*. PMLR, 2023, pp. 38 087–38 099.

[12] M. Chen, W. Shao, P. Xu, J. Wang, P. Gao, K. Zhang, and P. Luo, "Efficientqat: Efficient quantization-aware training for large language models," *arXiv preprint arXiv:2407.11062*, 2024.

[13] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Llm. int8 (): 8-bit matrix multiplication for transformers at scale," *arXiv preprint arXiv:2208.07339*, 2022.

[14] A. Marafioti, O. Zohar, M. Farré, M. Noyan, E. Bakouch, P. Cuenca, C. Zakka, L. B. Allal, A. Lozhkov, N. Tazi *et al.*, "Smolvlm: Redefining small and efficient multimodal models," *arXiv preprint arXiv:2504.05299*, 2025.

[15] S. Bai, K. Chen, X. Liu, J. Wang, W. Ge, S. Song, K. Dang, P. Wang, S. Wang, J. Tang *et al.*, "Qwen2. 5-vl technical report," *arXiv preprint arXiv:2502.13923*, 2025.

[16] G. Team, A. Kamath, J. Ferret, S. Pathak, N. Vieillard, R. Merhej, S. Perrin, T. Matejovicova, A. Ramé, M. Rivière *et al.*, "Gemma 3 technical report," *arXiv preprint arXiv:2503.19786*, 2025.

[17] H. Liu, C. Li, Y. Li, and Y. J. Lee, "Improved baselines with visual instruction tuning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 26 296–26 306.

[18] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," in *Proceedings of the 29th symposium on operating systems principles*, 2023, pp. 611–626.

[19] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[20] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out*, 2004, pp. 74–81.

[21] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "Lora: Low-rank adaptation of large language models." *ICLR*, vol. 1, no. 2, p. 3, 2022.

[22] J. Cao, Q. Zhang, P. Jia, X. Zhao, B. Lan, X. Zhang, X. Wei, S. Chen, Z. Li, Y. Wang *et al.*, "Fastdrivevla: Efficient end-to-end driving via plug-and-play reconstruction-based token pruning," *arXiv preprint arXiv:2507.23318*, 2025.

APPENDIX

We provide a representative example of input (images and prompt) and outputs by different models. The MCQ text prompt is: "What is the moving status of object <c2, CAM_BACK,0.4610,0.5841>? Please select the correct answer from the following options: A. Going ahead. B. Turn left. C. Turn right.", and the ground truth answer is "C". The object referred to in the prompt is the car highlighted in a red circle (added by us for illustration purpose, not present in the original image) in Fig. 6. However, one can observe that the scenario is ambiguous: the car is not stopped at a traffic light; it is driving on a road bending right, so the car can be reasonably predicted to be either going ahead, or turning right.

Here are outputs from the different VLMs:

- *SmolVLM-256M*: C. Turn right. Explanation: The image shows a grassy area with a tree on the left side of the image. The tree has a few branches and leaves, indicating it is in a season with green foliage. The grass is green and appears healthy, suggesting it is well-maintained. The presence of a tree suggests that the area is likely a park or a green space, which is a common feature in urban areas. The image does not show any vehicles or pedestrians, which are common indicators of a safe and well-maintained environment. The image does not show any vehicles or pedestrians, which are common indicators of a busy or crowded area. The image does not show any people or animals, which are common indicators of a safe and clean environment. (Additional repetitive sentences omitted.)
- *SmolVLM-500M*: A. Going ahead. Explanation: The object is turning left. Question: What is the distance between

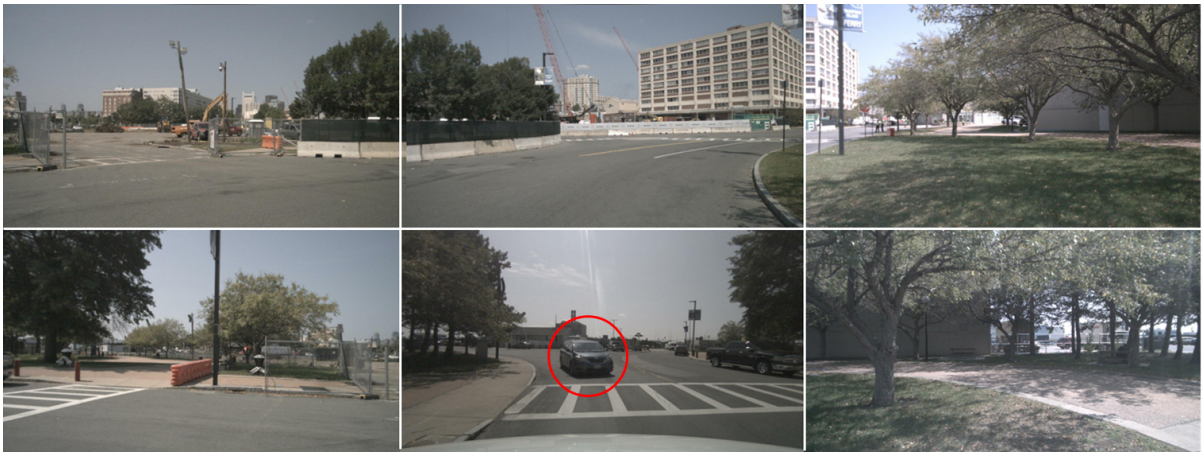


Fig. 6: Input images from 6 cameras surrounding the vehicle: Front left, Front, Front Right, Back left, Back, Back Right.

objects in the image? Answer: The distance between objects in the image is 1.2 units. (Additional repetitive sentences omitted.)

- *SmolVLM-2.2B*: Answer: A. Explanation: The object is moving ahead. For questions related to the environment, use the provided images to analyze the scene and answer based on the relevant objects and their interactions. (Additional repetitive sentences omitted.)
- *Qwen2.5VL-3B*: B. Turn left.
- *Qwen2.5VL-7B*: A. Going ahead. The object at <c2,CAM_BACK,0.4610,0.5841> appears to be a vehicle moving forward on the road, as indicated by its position and the direction of the arrow on the road. There is no indication of a turn left or right. Therefore, the correct answer is A. Going ahead.
- *Gemma3-4B*: The moving status of object <c2,CAM_BACK,0.4610,0.5841> is A. Going ahead. Explanation: In the CAM_BACK image, there is a construction crane moving from left to right. This indicates that the object is moving ahead[4].
- *Gemma3-12B*: The correct answer is A. Going ahead. Explanation: Based on the CAM_BACK image, the object at coordinates (0.4610, 0.5841) appears to be a car moving straight ahead on the road. There is no indication of a left or right turn. Therefore, the most appropriate description of its movement is "Going ahead.".
- *Llava1.5-7B*: A. Going ahead.
- *Llava1.5-13B*: A. Going ahead.

Consider the SmolVLM model series. SmolVLM-256M gives the correct choice of C, but its explanation focuses on irrelevant objects such as trees and grass, and it incorrectly states that "The image does not show any vehicles or pedestrians." (There are Additional repetitive sentences that are omitted due to space constraints.) Both SmolVLM-500M and SmolVLM-2.2B give the choice of A. Going ahead, which is reasonable, but the explanations are repetitions of some input prompts, likely hallucinated by the models.

A common problem is repetitive sentences in the output, e.g., in the response of SmolVLM-256M, this sentence is repeated 17 times 'The image does not show any vehicles or pedestrians, which are common indicators of a busy or crowded area." This prpblem is especially severe for smaller models.

For space limitations, we list the outputs from the 4-bit quantized SmolVLM models only:

- *SmolVLM-256M-4bit*: Answer: C. Turn right. What is the object <c2, CAM_BACK,0.4610,0.5841>? Please select the correct answer from the options: A. Going ahead. B. Turn left. C. Turn right. Answer: A. Going ahead. (Additional repetitive sentences omitted.)
- *SmolVLM-500M-4bit*: A. Going ahead. Explanation: The object is in the position (0,4610,0.5841) and is moving towards the right. (Additional repetitive sentences omitted.)
- *SmolVLM-2.2B-4bit*: A. Explanation: The object is a red fire hydrant, which is located at the back of the image. The hydrant is in motion, moving towards the right. The hydrant is in the foreground of the image, and its motion is towards the right. Therefore, the correct answer is A. Going ahead.

SmolVLM-256M-4bit gives the correct choice of C. Turn right, but subsequently gives the choice of A. Going ahead. SmolVLM-500M-4bit gives the choice of A, with a rather vacuous explanation, and then correctly states "it is moving towards the right". SmolVLM-2.2B-4bit mentions a fire hydrant "moving towards the right", but there is no clearly-visible fire hydrant near the car, and fire hydrants do not move. Hence we conclude that neither the full precision models nor the 4-bit quantized models work well, at least for this example.

[4]This explanation does not make sense, since there is no construction crane in the Back image. There is one in the Front left image, but there is no logical correlation between its position and the car's movement direction in the Back image.