Hardware Solutions for Vision Language Models in Autonomous Driving

Luigi Altamura
Supervised by Pedro Trancoso and Mohammad Ali Maleki
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden
altamura@chalmers.se

Abstract

End-to-end deep learning systems have become central to autonomous driving, outperforming modular architectures in many benchmarks. However, their black-box nature limits transparency and explainability, which are key requirements in safety-critical environments. Recently, Vision Language Models (VLMs) have emerged as a promising solution, offering natural language explanations of driving decisions. However, their high computational and memory demands present major challenges for real-time deployment in vehicles. This work explores hardware-oriented solutions to enable VLMs in autonomous driving, with a focus on energy efficiency, latency, and scalability. It investigates accelerator technologies capable of supporting VLM deployment under strict automotive constraints. The goal is to assess and guide the development of hardware platforms that can make explainable AI feasible for next-generation autonomous vehicles.

CCS Concepts

• **Computer systems organization → Architectures**.

Keywords

Hardware Accelerators, Autonomous Driving, Vision Language Models, Explainable AI

ACM Reference Format:
Luigi Altamura. 2025. Hardware Solutions for Vision Language Models in Autonomous Driving. In *Proceedings of 19TH ACM INTERNATIONAL CONFERENCE ON DISTRIBUTED AND EVENT-BASED SYSTEMS (DEBS '25).* ACM, New York, NY, USA, 4 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Problem Statement

Autonomous driving (AD) has seen rapid progress in recent years, largely driven by advances in Artificial Intelligence (AI). Two primary paradigms have emerged in this space: modular pipelines and end-to-end learning approaches [14]. Modular systems break down the driving task into several stages like perception, planning, and control, offering better explainability and easier debugging. However, they often suffer from performance bottlenecks and error accumulation across stages [12]. In contrast, end-to-end models use deep neural networks to directly map sensor inputs to driving commands. These models have shown strong performance in complex scenarios, but their black-box nature limits transparency, an essential feature for safety-critical applications such as AD.

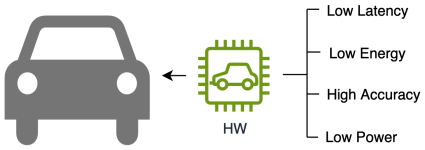


Figure 1: Accelerator requirements overview

A promising direction for improving the explainability of end-to-end models in AD is the use of Vision Language Models (VLMs). These models combine visual perception with natural language reasoning, enabling them to produce human-readable explanations for their driving decisions. Figure 2 shows a general VLM architecture, which consists of a vision encoder whose output is mapped by a multimodal projector into embeddings that, together with the text prompt, are fed into an LLM to generate a text answer.

This added explainability makes them especially appealing for safety-critical systems, where understanding the decision-making process is as important as achieving high performance.

VLMs have already demonstrated strong results in controlled environments. For example, CarLLaVA [9], a VLM designed for autonomous driving, currently ranks among the top-performing systems on the CARLA Leaderboard [2], showcasing the potential of VLMs to lead in both performance and explainability when evaluated in high-fidelity simulators like CARLA [4].

However, this capability comes at a significant computational cost. VLMs are resource-intensive, requiring substantial compute and memory to process multimodal inputs and generate meaningful outputs. These demands pose serious challenges for real-time inference on embedded automotive platforms [8], where constraints on latency, energy consumption, power, and hardware capacity are much tighter than in cloud or simulation environments, as shown in Figure 1.

This research aims to address this gap by exploring hardware-level solutions for running VLMs efficiently in autonomous driving settings.

DEBS '25, Gothenburg, SE

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

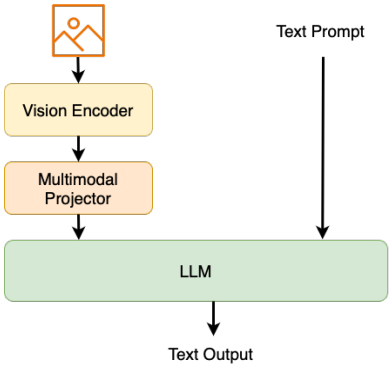


Figure 2: General VLM architecture

2 State of the Art

Optimizing the execution of LLMs and VLMs is an active and fast-moving area of research. The focus of this research is to identify hardware-level strategies that enable the practical deployment of VLMs in autonomous vehicles.

At present, there is no established state-of-the-art hardware architecture specifically designed for VLMs, either in general or within the context of autonomous driving. This is probably because most VLMs adopt a vision encoder whose structure is similar in complexity to the language model backend, inheriting many of the same bottlenecks as traditional LLMs. As a result, most deployment efforts rely on accelerators originally optimized for generic LLMs, which may not meet the unique demands of vision-language reasoning under real-time constraints.

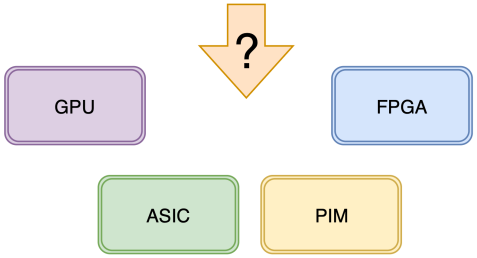
It is important to note that optimizing VLMs presents a different set of challenges compared to other models for AD like Transfuser [3] or Interfuser [10]. VLMs typically involve larger model sizes and significantly higher computational loads due to the language reasoning component and complex cross-modal interactions. As shown in Table 1, VLMs exhibit a notably higher number of parameters and operations, justifying the need for dedicated hardware exploration and tailored acceleration strategies.

Existing hardware accelerators for LLMs fall into several categories [7]:

- **GPUs** are the dominant platform due to their high parallelism, support for mixed-precision arithmetic, and robust software ecosystems. However, they often fail to meet the energy requirements of embedded automotive use cases.
- **FPGAs** offer reconfigurability and improved energy efficiency. Solutions like FlexRun [5] demonstrate the viability of LLM inference on custom pipelines with quantized operators, and include comparisons against models like GPT2. However, their performance on VLMs remains largely unexplored.
- **ASICs** are increasingly used in automotive applications for their ability to deliver dedicated acceleration for specific workloads. One example is Hailo [1], which offers ASIC-based solutions for tasks such as ADAS and perception. However, these systems are not currently designed to handle the computational demands of fully autonomous driving or the heavy multimodal inference required by VLMs.
- **Processing-In-Memory (PIM)** architectures are gaining attention as a promising direction for future deployment. They reduce data movement by performing computation close to memory, improving energy efficiency for large-scale models. For example, TransPIM [15] demonstrates the potential of combining PIM and Near-Memory Computing to significantly reduce data movement and improve efficiency in large model inference. However, such approaches have not yet been tested in the context of autonomous driving, and their suitability for real-time VLM workloads in this domain remains an open question.

Table 1: Comparison of VLM and end-to-end models in terms of complexity and explainability.

Model	Parameters	MACs	Explainability
Transfuser	++	+	✗
Interfuser	+	+	✗
VLLMs	+++	+++	✓



In summary, the field lacks a clear reference architecture for VLM deployment, particularly in constrained environments like autonomous vehicles. As highlighted in Table 2, no existing accelerator type simultaneously satisfies all the key requirements, and proven automotive applicability. This research aims to explore and evaluate hardware accelerators—including conventional and emerging technologies—capable of meeting the latency, power, and accuracy demands of VLMs, while remaining scalable and cost-effective for future real-world integration.

3 Methodology

This work uses a hardware-focused methodology to understand how VLMs behave under the strict real-time and energy constraints of autonomous driving. The main goal is to collect insights that will guide the design of a custom hardware accelerator suited for running VLMs efficiently in this setting. By identifying where current models and hardware struggle — such as high latency, memory bottlenecks, or energy usage — we can define what the accelerator needs to handle.

3.1 Layer-Wise Analysis

To optimize VLMs for real-time autonomous driving, it is essential to understand which parts of the model contribute most to computational load and latency. A layer-wise analysis enables this by

Table 2: Comparison of accelerator types across key characteristics relevant to VLM deployment in AD.

Accelerator	Reconfigurable	Energy Efficient	VLM Ready	Automotive Proven
GPU	✗	✗	✓	✗
FPGA	✓	✓	✗	✗
ASIC	✗	✓	✗	✗
PIM	✗	✓	✗	✗

breaking down the model into individual components and identifying "hot points" — layers or blocks that are expensive in terms of computation, memory, or energy usage. This analysis supports targeted optimizations where they will have the highest impact. Specifically, we measure:

- **FLOPs and parameter counts per layer** to assess compute demand and memory storage needs. FLOPs reveal where peak compute occurs, while parameter counts reflect the model's memory footprint—critical for fitting models on memory-constrained devices.
- **Separate profiling of the visual encoder and language model components** to highlight key differences in latency, compute intensity, and memory usage. This helps determine which part is the primary performance bottleneck and guides hardware-aware optimizations.

This granular analysis helps prioritize which layers or components are best suited for approximation, fusion, or acceleration, and informs targeted hardware optimizations.

3.2 Hardware Profiling and Roofline Modeling

To systematically quantify hardware limitations, we employ **roofline modeling** [6, 13], a method to evaluate practical performance bounds of hardware platforms. The roofline model helps distinguish whether a given model or layer is:

- **Compute-bound** — limited by peak FLOP/s of the hardware, or
- **Memory-bound** — limited by memory bandwidth and data movement overheads.

By plotting operational intensity against achieved throughput, we can assess how efficiently the hardware is utilized. This modeling will be applied across multiple devices (e.g., GPUs, edge accelerators) to evaluate:

- Hardware utilization gaps—how far actual performance is from the roofline.
- The impact of memory access patterns, particularly in attention mechanisms and vision transformers, on bandwidth constraints.

Roofline modeling provides the foundation for justifying architectural decisions such as operator fusion, tiling, memory layout transformations, and deployment on specialized hardware.

3.3 Hardware Design and Simulation

Based on profiling insights, this research will explore the design of a custom accelerator tailored for VLM workloads under automotive constraints. The focus is on minimizing latency and energy consumption while supporting the memory and compute demands of vision and language components.

Simulations will be used to evaluate key design metrics such as area, performance, and energy efficiency. Particular attention will be given to memory hierarchies, data movement, and the mapping of compute-intensive operations like attention and matrix multiplications. The goal is to identify architectural features that make accelerators better suited for real-time VLM execution in autonomous vehicles.

4 Research setting

Deploying VLMs for autonomous driving requires a tailored research setup, significantly different from their traditional use in general-purpose language or vision-language tasks. This section outlines the specific constraints and considerations involved.

4.1 Real-Time Inference Constraints

Autonomous driving systems must operate in real time, where delays can compromise safety and performance. To meet these demands:

- Batch size must be fixed to 1, reducing latency by processing data frame-by-frame.
- A high frame rate is essential. Industrial systems (e.g., Tesla's) reportedly support up to 2030 fps [11], setting a high-performance reference.
- The system must support low-latency inference, compatible with the control loop of the vehicle.

4.2 Structured and Bounded Input/Output

Unlike open-ended applications of LLMs, this setting requires strict control over token usage. Input tokens consist of:

- A visual embedding (from camera image)
- Textual metadata (e.g., current speed, steering angle).
- A prompt requesting a concise explanation for the driving decision.

Output is expected to be limited in length compared to general-purpose VLM usage, focusing on concise explanations that effectively describe the driving scene.

This structured format ensures predictable memory and compute requirements, facilitating system-level optimization.

4.3 Hardware and Energy Considerations

Since the model runs on vehicle-embedded hardware, additional constraints apply:

- Energy efficiency is critical due to reliance on the vehicle's battery power.
- The hardware cost must be reasonable; accelerators should not rival the cost of the car itself.

This research will explore:

- How to adapt VLMs to this structured, real-time setting.
- Strategies to optimize latency, power, energy consumption, and hardware cost.
- The feasibility of deploying explainable AI models in real-world autonomous systems while preserving practical performance standards.

5 Research approach

This research proposes a hardware-centered strategy to make VLMs suitable for real-time autonomous driving. The central idea is to retain the explainability and flexibility of these models while optimizing their execution for embedded, battery-powered systems with strict latency and energy constraints.

The proposed solution focuses on adapting VLMs to operate under a constrained inference setup: single-frame (batch size = 1) processing, limited token budgets for both input (image + driving state) and output (natural language explanation), and consistent, bounded compute requirements. By limiting the input/output size and analyzing inference patterns, the model's performance can be made more predictable and efficient.

To address the research questions, this approach investigates:

- Whether the high computational cost of VLMs can be reduced without sacrificing the explainability benefits.
- How the hardware architecture (e.g., memory bandwidth, accelerator capabilities) influences real-time performance.
- Which optimizations (e.g. fixed token lengths) are most effective under real-world constraints.

The aim is not to refine models, but to enable their practical deployment by aligning them with the requirements of embedded automotive systems.

6 Evaluation plan

The evaluation will focus on the hardware feasibility and computational efficiency of deploying VLMs in real-time driving contexts. The key goal is to determine whether these models can meet the stringent latency, throughput, and energy constraints required for deployment in embedded automotive systems.

Key Evaluation Criteria:

- **Inference Latency:** Measuring end-to-end latency for single-frame processing (batch size = 1), across different hardware configurations.
- **Frame Rate (FPS):** Targeting a frame rate that meets real-time performance requirements in dynamic and safety-critical environments.
- **Energy Consumption:** Quantifying power draw, with a focus on feasibility for battery-powered embedded devices.
- **Memory Bandwidth and Utilization:** Profiling how efficiently models utilize available memory.
- **Hardware Cost and Practicality:** Evaluating the trade-off between performance gains and the physical/financial cost of specialized accelerators—ensuring the compute platform remains viable for in-vehicle deployment.

Model components (e.g., visual encoder vs. language decoder) will be analyzed independently to identify computational hotspots and guide potential optimizations such as quantization, KV caching, and token-length control.

7 Conclusions and reflections

This research is in its early stages, with the problem and methodology defined but no experimental results yet. The focus is on enabling real-time deployment of VLMs under strict hardware constraints, balancing accuracy, latency, and energy use.

A major challenge lies in the rapid evolution of both models and hardware, making adaptability a core requirement. The goal is to develop strategies that not only work today but remain viable as models grow in complexity. Despite the challenges, the potential to combine explainability with real-time performance makes this a promising and timely direction.

References

[1] Adas and ad solutions – hailo. https://hailo.ai/applications/automotive/adas-and-ad/#adas-overview, 2025. Accessed: May 21, 2025.

[2] Carla autonomous driving leaderboard, 2025. Accessed: May 20, 2025.

[3] Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving, 2022.

[4] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator, 2017.

[5] Suyeon Hur, Seongmin Na, Dongup Kwon, Joonsung Kim, Andrew Boutros, Eriko Nurvitadhi, and Jangwoo Kim. A fast and flexible fpga-based accelerator for natural language processing neural networks. *ACM Trans. Archit. Code Optim.*, 20(1), February 2023.

[6] Aleksandar Ilic, Frederico Pratas, and Leonel Sousa. Cache-aware roofline model: Upgrading the loft. *IEEE Computer Architecture Letters*, 13(1):21–24, 2013.

[7] Jinhao Li, Jiaming Xu, Shan Huang, Yonghua Chen, Wen Li, Jun Liu, Yaoxiu Lian, Jiayi Pan, Li Ding, Hao Zhou, Yu Wang, and Guohao Dai. Large language model inference acceleration: A comprehensive hardware perspective, 2025.

[8] Katrin Renz, Long Chen, Elahe Arani, and Oleg Sinavski. Simlingo: Vision-only closed-loop autonomous driving with language-action alignment, 2025.

[9] Katrin Renz, Long Chen, Ana-Maria Marcu, Jan Hünermann, Benoit Hanotte, Alice Karnsund, Jamie Shotton, Elahe Arani, and Oleg Sinavski. Carllava: Vision language models for camera-only closed-loop driving, 2024.

[10] Hao Shao, Letian Wang, RuoBing Chen, Hongsheng Li, and Yu Liu. Safety-enhanced autonomous driving using interpretable sensor fusion transformer, 2022.

[11] Emil Talpes, Debjit Das Sarma, Ganesh Venkataramanan, Peter Bannon, Bill McGee, Benjamin Floering, Ankit Jalote, Christopher Hsiong, Sahil Arora, Atchyuth Gorti, and Gagandeep S. Sachdev. Compute solution for tesla's full self-driving computer. *IEEE Micro*, 40(2):25–35, 2020.

[12] Xu Wang, Mohammad Ali Maleki, Muhammad Waqar Azhar, and Pedro Trancoso. Moving forward: A review of autonomous driving software and hardware systems, 2024.

[13] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Commun. ACM*, 52(4):65–76, April 2009.

[14] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020.

[15] Minxuan Zhou, Weihong Xu, Jaeyoung Kang, and Tajana Rosing. Transpim: A memory-based acceleration via software-hardware co-design for transformer. *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA).*