Time-constrained Vision Language Model Inference using Semantic Caching for Autonomous Driving

Manuel Branco Nardi and Chanhee Lee

University of Central Florida. {ma352154, chanhee.lee}@ucf.edu

***Abstract*—Vision-language models (VLMs) have shown strong potential for understanding complex driving environments, but their high inference latency remains challenging under real-time constraints. To address this challenge, we propose a semantic caching-based VLM inference for autonomous driving context analysis. The key idea is to identify semantically similar road scenes and reuse the VLM outputs of the previously stored scenes. Our approach integrates CLIP-based image embeddings and a vector database into the Lingua Franca (LF) reactor model to ensure time constraints are met. Evaluation with three VLMs demonstrates that the proposed approach achieves an average inference throughput 10.5 times higher than VLM inference without caching while maintaining high inference accuracy across various driving-scene types.**

***Index Terms*—Autonomous driving, Vision Language Model, Lingua Franca**

I. Introduction

Introducing Vision Language Models (VLMs) to autonomous driving in urban environments has recently gained significant interest from researchers and industry due to their revolutionary ability to recognize human-like scene understanding [1]–[5]. Autonomous driving involves various CPS technologies, such as real-time image processing and sensor management, as well as runtime inference and control invocations to properly drive a vehicle. All these executions should be completed within the given time constraints to ensure safe, critical vehicle operations.

Some existing approaches [1], [2], [4] assess that VLM inference latency remains a challenge for real-time deployment of VLMs in autonomous driving systems. To clarify the degree of the current VLM inference latencies, we analyze the average latencies of three VLMs, Qwen2.5-VL [6], Gemma3 [7], and LLaVA-Phi3 [8], as shown in Fig. 1. The VLM inference latencies range from 17.5 to 36.0 s per image with GPU acceleration, while the range becomes even worse, ranging from 44.9 to 115.1 s with CPU-only inference. The results clearly confirm that VLM inference necessitates performance optimization to meet real-time constraints.

During the latency evaluation, we have found that there are plenty of redundant inferences that produce the same or similar VLM outputs. This is because consecutive camera frames in real-world driving often capture highly similar scenes, and vehicles frequently encounter recurring road configurations and traffic situations across different locations. Introducing an efficient caching structure for input images during VLM inference can enable the reuse of previously generated VLM outputs, such as scene descriptions, thereby treating parts of consecutive VLM inferences as redundant and skipping them.

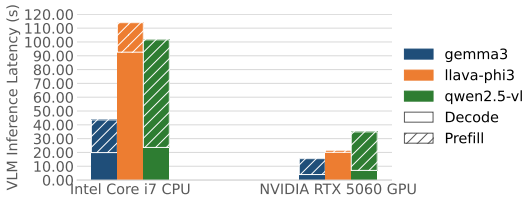

Fig. 1: The latency breakdown results with three VLMs.

Therefore, based on this insight, we propose a semantic caching approach that leverages Contrastive Language–Image Pretraining (CLIP) [9] embeddings and a vector database to identify similar scenes that have been previously analyzed and stored. Then, we use a similarity metric to retrieve previously cached VLM inference outputs, dynamically adjusting the similarity threshold to improve VLM inference throughput while maintaining VLM output accuracy. The computational cost of consecutive VLM inferences that produce the same outputs can be significantly reduced by forwarding only scenes that cannot be matched to existing cache entries to the VLM for analysis.

Furthermore, to bound the VLM inference latency deterministically considering time constraints, we integrate Lingua Franca (LF) [10] reactor model. LF provides an event-driven decomposition of the perception pipeline, allowing the system to be implemented as modular components with explicit communication paths and timing semantics. Through experiments with a calibrated prompt, large autonomous driving datasets, and multiple VLMs, we demonstrate that our approach achieves an average VLM inference throughput 10.5 times higher. The contributions of this work can be summarized as follows.

- We present a semantic caching-based VLM inference for autonomous driving that dynamically adjusts similarity to improve the inference throughput.
- We implement the proposed approach using Lingua Franca, enabling reactor-based time-constraint assurance of VLM inferences.
- The evaluation with the latest VLM dataset demonstrates that our approach improves VLM inference throughput 10.5 times on average while maintaining VLM accuracy.

II. Background

Vision Language Model. Vision-language models (VLMs) extend large language models (LLMs) to process both images and text, enabling models to generate outputs based on visual and textual inputs. They are widely used for tasks such as image captioning, visual question answering, and multimodal reasoning. Modern VLMs generally combine a vision encoder with a language model, together with additional components that allow visual information extracted from images to be incorporated into the language model's processing.

Semantic Caching. Semantic caching [11] is a method for retrieving previously computed outputs and their inputs to reuse the outputs for new inputs that are similar to those already stored. Inputs are stored as fixed-size vectors via an embedding and compared with the embeddings of new inputs using a similarity metric. For example, in each VLM inference with an incoming image I_i, the semantic caching performs nearest-neighbor retrieval by computing the similarity score, such as cosine similarity, between the incoming frame embedding and the closest cached embedding of image I_c. A cache hit occurs when $S(I_i, I_c) \geq \tau$ where τ is the similarity threshold and $S(I_i, I_c)$ is the similarity score between the current frame and some previous stored frame in the database.

Lingua Franca. Lingua Franca (LF) is a coordination language designed to simplify the construction of time-sensitive and concurrent cyber-physical systems. Instead of relying on low-level concurrency mechanisms or middleware callbacks, LF structures applications around reactors, which are concurrent components that react to events in a deterministic order. Each reactor encapsulates its internal state, input and output ports, and a set of reactions. LF also supports explicit deadlines, enabling detection and handling of timing violations.

III. Design

A. Semantic Caching-based VLM Inference

The proposed approach uses semantic caching to identify redundant inferences based on a similarity score. The VLM inference outputs of previously analyzed and stored scenes are extracted when the similarity score between the current input and the stored scene meets the current similarity threshold. The core of our design is to dynamically adjust the similarity threshold to meet both the target VLM output accuracy and the target throughput at runtime. The lightweight, efficient threshold adjustment at run-time is important because the trade-off between VLM throughput and accuracy should be considered jointly.

Fig. 2 demonstrates the trade-off relationship between accuracy and throughput across different thresholds. When the similarity threshold is set to 1.0, the VLM inference throughput, in frames per second (FPS), is the lowest at 0.3 FPS. Only exactly matched previous scenes can be reused, and VLM inference omissions occur less often. However, in this case, the VLM achieves a higher accuracy of around 0.45 because the two images being compared are very similar. On the other hand, if the threshold is set below 0.75, the accuracy of the reused VLM output drops significantly, falling below 0.4.

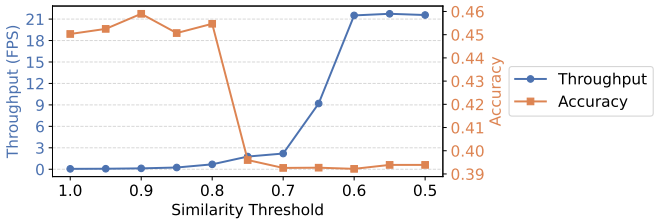

Fig. 2: Trade-off analysis of throughput and accuracy.

Algorithm 1 Similarity Threshold Adjustment

1: Input: similarity threshold step Δs, minimum accuracy A_{min}, target throughput T_{target}
2: $\tau \leftarrow 1.0$
3: **while** $\tau \geq 0$ **do**
4: Estimate accuracy $A(\tau)$ and throughput $T(\tau)$
5: **if** $A(\tau) < A_{min}$ **return** previous threshold
6: **if** $T(\tau) \geq T_{target}$ **return** τ
7: $\tau \leftarrow \tau - \Delta s$
8: **end while**

Algorithm 1 presents the proposed heuristic for determining a similarity threshold that satisfies the target VLM inference throughput and minimum VLM inference output accuracy specified (line 1). The inputs are application-dependent parameters selected by the system designer. Starting from $\tau = 1.0$ (line 2), the similarity threshold is iteratively reduced by the step size Δs (line 7) until it reaches 0 (line 3). The system throughput and accuracy are re-evaluated after each threshold adjustment (line 4). The procedure terminates when the measured accuracy falls below the specified minimum A_{min} (line 5), in which case the previous threshold is returned, or the target throughput T_{target} is achieved (line 6).

B. VLM Prompt for Autonomous Driving

```
You are a road-scene classifier for an autonomous
driving system. Analyze the image and return a
single JSON object. Use ONLY the exact values listed
below for each field. Return nothing else.
Scene: one of "Urban", "Suburban", ...
TimeOfDay: one of "Daytime", "Dusk/Dawn"
Weather: one of "Clear", "Sunny", "Cloudy", ...
RoadConditions: one of "Dry", "Wet"
LaneInformation:
  NumberOfLanes: one of "NoLane", "1", "2", "3", ...
  LaneMarkings: one of "LaneVisible", ...
TrafficSigns:
  TrafficSignsTypes: array | each item one of
"NoTrafficSigns", "Speed Limit", ...
Vehicles:
  TotalNumber: one of "NoVehicle", "One", ...
  VehicleTypes: array of vehicle types visible |
each item one of "Cars", "Sedan", ...
Pedestrians: array | each item one of "NoPed",
"Crossing Street", "Visible on Sidewalk", ...
Directionality: one of "One-Way", "Two-Way", ...
Visibility: General: one of "Good", "Limited", ...
CameraCondition: one of "Clear", "Dirty", ...
```

We design the VLM prompt as the above to generate structured scene descriptions using a predefined JSON schema aligned with the CARScenes annotation structure [12]. Rather

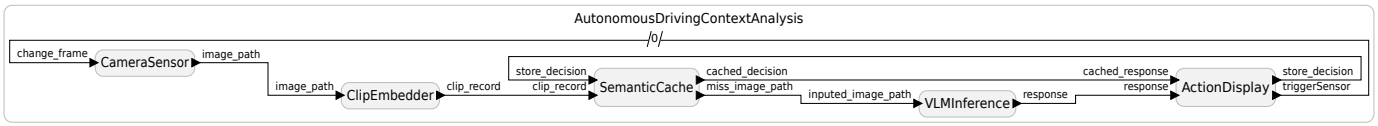

Fig. 3: The LF diagram of the proposed VLM inference architecture for autonomous driving.

than producing free-form natural language, the prompt constrains the model to select values from predefined semantic categories describing scene context, environmental conditions, lane information, traffic signs, vehicles, pedestrians, visibility conditions, ego-vehicle behavior, and overall scene severity. This representation enables direct comparison with CARScenes annotations and supports downstream retrieval of previously generated decisions from the vector database.

IV. Implementation

We implemented our semantic caching-based VLM inference architecture with five LF reactors, as shown in Fig. 3.

CameraSensor reactor periodically acquires image frames from the driving environment and forwards them to the embedding stage.

ClipEmbedder reactor uses the CLIP image encoder to generate normalized image embeddings representing the semantic content of each road scene. These embeddings provide compact feature representations suitable for similarity search.

SemanticCache reactor performs semantic retrieval using a persistent ChromaDB vector database [13]. For each incoming CLIP embedding, the reactor performs a nearest-neighbor search using cosine similarity. If the similarity exceeds a threshold determined by Algorithm 1, the corresponding cached driving decision, previously generated by the VLM and stored to the vector database, is returned from the vector database. Otherwise, the reactor forwards the image to the *VLMInference* reactor.

VLMInference reactor executes only when a cache miss from the vector database occurs. The reactor invokes the VLM to generate a structured driving decision containing hazard information, driving action recommendations, steering adjustments, and speed recommendations. LF deadlines are used to detect timing violations during inference execution.

ActionDisplay reactor receives either cached or newly generated VLM decisions and presents the results to downstream components. When a decision originates from the VLM, it is inserted into the semantic cache together with the associated image embedding, enabling future reuse.

The reactor-based implementation using LF enables precise timing analysis under the given time constraint. To maintain responsiveness, the proposed approach executes VLM inference asynchronously. When a cache miss occurs, inference is launched in a separate execution context, allowing the reactor to immediately return control to the LF runtime. A polling mechanism periodically checks for completion and forwards results once they become available. This approach prevents long-running computations from blocking logical time progression.

The given deadline is configured within *VLMInference* reactor using LF semantics to prevent logical time from lagging excessively behind physical time. Asynchronous inference and periodic polling detect deadline violations precisely with nanosecond accuracy by ensuring that long-running VLM computations do not block other reactors. LF integrates timing constraints directly into the reactor model, allowing timing behavior to be specified alongside application logic. It also enables accurate and efficient implementation for deadline monitoring without introducing other external timing mechanisms.

V. Experimental Results

A. Setup

We run the experiments on the system described in Table I. As VLMs, we use the three open-source vision language models: Qwen2.5-VL, Gemma3, and LLaVA-Phi3, which have 7B parameters and 6 GB in size, 4B parameters and 3.3 GB in size, and 3.8B parameters and 2.9 GB in size, respectively. As input driving scenes for the VLMs, we use multiple driving scenarios from the CARScenes dataset, which is derived from the NuScenes dataset [14]. The frame sequences from the dataset emulate a forward-facing vehicle camera stream and capture common urban driving situations, including pedestrian crossings, vehicles entering traffic lanes, intersections, and other road conditions. Consecutive frames within a scenario often exhibit only minor visual changes, making them representative of real-world driving environments where adjacent camera frames are highly correlated. To estimate VLM accuracy, we compare the generated outputs against the corresponding CARScenes annotations using BLEU metrics [15].

B. Performance Analysis

Throughput. In the Qwen2.5-VL results, our approach performs 118 VLM inferences on 1262 input images, indicating a 10.5 times improvement in VLM inference throughput. 90.6% of the total VLM inference can be omitted by reusing the results stored in the vector database through the proposed semantic caching approach. Similar cache reuse rates were observed for Gemma3 (90.73%) and LLaVA-Phi3 (91.13%). These results demonstrate that semantic caching substantially reduces the computational cost of applying VLMs to long driving sequences.

TABLE I: Experimental execution environment.

CPU	Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz
Memory	32GB DDR5
OS and Kernel	64-bit Ubuntu 24.04 LTS; 6.17.0-14-generic
GPU	NVIDIA GeForce RTX 5060, 24GB VRAM

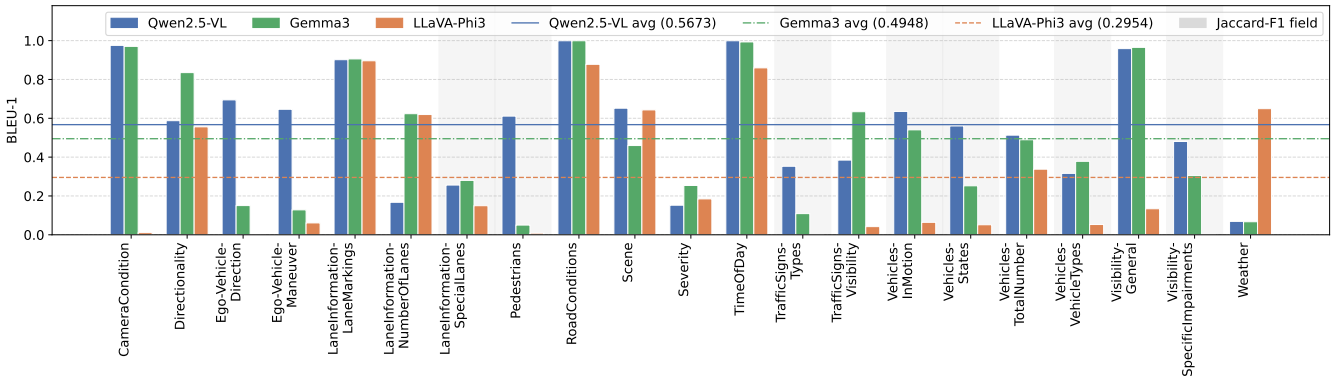

Fig. 4: The detailed accuracy results for various driving contexts with the three VLMs.

TABLE II: Inference output accuracies of the three VLMs on the CARScenes dataset.

Model	Macro Avg.	BLEU-1	BLEU-2	BLEU-3	BLEU-4	Corpus BLEU
Qwen2.5-VL	**0.5673**	**0.5432**	**0.2606**	**0.1297**	**0.0661**	**0.1867**
Gemma3	0.4948	0.3817	0.1440	0.0519	0.0200	0.0869
LLaVA-Phi3	0.2954	0.0548	0.0271	0.0182	0.0120	0.0239

Accuracy. Among the evaluated models, Qwen2.5-VL achieved the highest macro-average score[1] (0.5673) and Corpus BLEU score (0.1867), as shown in Table II. Gemma3 achieved the second-highest macro-average (0.4948) and BLEU (0.0869), while LLaVA-Phi3 obtained the lowest scores (0.2954 and 0.0239), respectively. This is because the Qwen model has the most parameters, while the LLaVA-Phi3 has the fewest. Fig. 4 presents the detailed accuracy results of the Qwen model for driving semantics defined in the VLM prompt in Section III-B. Although Qwen2.5-VL achieved the highest overall accuracy, Gemma3 remained competitive despite fewer parameters, outperforming Qwen2.5-VL in Directionality (0.8352 vs. 0.58), LaneInformation.NumberOfLanes (0.6236 vs. 0.17), and TrafficSigns.Visibility (0.6338 vs. 0.38). In contrast, Qwen2.5-VL achieved substantially higher scores in categories related to ego-vehicle behavior and pedestrian understanding. These results suggest that while larger VLMs generally provide stronger overall scene understanding, smaller models can achieve competitive performance for specific driving-scene attributes at a lower computational cost.

VI. Conclusion

This paper presented a semantic caching VLM inference for autonomous driving perception using Lingua Franca reactors, CLIP embeddings, and a vector database. Experimental evaluation with three VLMs on the CARScenes dataset demonstrates that the VLMs can generate structured semantic descriptions of driving scenes correctly, improving the throughput significantly by an 10.5 times, achieving an average accuracy score of 0.5673.

[1]A mismatch between the model and the label does not necessarily indicate an incorrect description, as multiple valid semantic descriptions may exist for the same driving scene.

References

[1] Y. Zhou, C. Cui, J. Peng, Z. Yang, J. Lu, J. Panchal, B. Yao, and Z. Wang, "A hierarchical test platform for vision language model (vlm)-integrated real-world autonomous driving," *ACM Transactions on Internet of Things*, 2025.

[2] R. Zhao, Q. Yuan, J. Li, Z. Wang, Y. Li, Z. Gao, H. Hu, and F. Gao, "Vlm-driver: Human-like autonomous driving decision-making via vision language model," *IEEE Transactions on Vehicular Technology*, 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.

[4] Y. Fu, A. Jain, X. Chen, Z. Mo, and X. Di, "Drivegenvlm: Real-world video generation for vision language model based autonomous driving," in *2024 IEEE International automated vehicle validation conference (IAVVC)*. IEEE, 2024, pp. 1–6.

[5] H. Tian, K. Reddy, Y. Feng, M. Quddus, Y. Demiris, and P. Angeloudis, "Large (vision) language models for autonomous vehicles: Current trends and future directions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 27, no. 1, pp. 187–210, 2025.

[6] Qwen Team, "Qwen2.5 technical report," *arXiv preprint arXiv:2501.12220*, 2025. [Online]. Available: https://arxiv.org/abs/2501.12220

[7] 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.

[8] M. I. Abdin *et al.*, "Phi-3 technical report: A highly capable language model locally on your phone," Tech. Rep., Aug. 2024. [Online]. Available: https://arxiv.org/abs/2404.14219

[9] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[10] Lingua Franca Project, "Lingua franca documentation," https://www.lf-lang.org/docs/.

[11] S. Selvam, R. K. Rajendran, M. Sankaradas, A. Raghunathan, and S. T. Chakradhar, "Simcache: similarity caching for efficient vlm-based scene understanding," in *CVPR*, 2025, pp. 3327–3336.

[12] Y. He and W. Shi, "Carscenes: Semantic vlm dataset for safe autonomous driving," *arXiv preprint arXiv:2511.10701*, 2025.

[13] Chroma, "Chroma: The ai-native open source embedding database," https://www.trychroma.com, 2024.

[14] 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 *CVPR*, 2020, pp. 11 621–11 631.

[15] 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.