On-Demand Vehicular Service Deployment in 6G: A Collaborative Large-Small LM Architecture

Amine Kidane Ghebreziabiher, Gordon Owusu Boateng, Daniel Ayepah-Mensah, Rabeb Mizouni, Azzam Mourad, Hadi Otrok, Jamal Bentahar, and Sami Muhaidat

Large Language Models (LLMs) offer significant potential for enabling on-demand service deployment in Intelligent Transportation Systems (ITS). However, their burgeoning size and high computational requirements limit their feasibility in 6G vehicular networks. To address these challenges, this article presents L2S-LM, a novel collaborative LLM-Small Language Model (SLM) architecture for on-demand vehicular service deployment in 6G networks. The architecture follows a modular Perception–Prediction–Placement pipeline, distributing tasks across cloud, edge, and end layers. For perception, a cloud-based Multimodal LLM (MLLM) generates semantic scene representations as model inferences, which a lightweight edge-deployed SLM interprets to predict appropriate on-demand services for vehicular traffic events. To enhance efficiency, a memory augmentation mechanism retrieves relevant historical predictions, reducing redundant perception computations. Finally, the placement module employs LM-assisted optimization to select the best node and allocate resources for service deployment. Experimental results demonstrate that L2S-LM achieves performance comparable to cloud-based LLM-only architecture while minimizing resource consumption.

Introduction

The rapid advancement of Intelligent Transportation Systems (ITS) as a core technology in the forthcoming Sixth-Generation (6G) wireless network paradigm is driving a shift toward real-time, context-aware, and on-demand vehicular service deployment. On-demand vehicular service deployment refers to dynamically adjusting the placement of vehicular network services in response to sudden traffic events, environmental variations, and user-specific demands [1]. This extends beyond existing static, pre-planned approaches, as the dynamism and customization of vehicular networks necessitate deploying tailored services to address various vehicular traffic events. For instance, on-demand emergency response service is crucial in the event of accidents, while real-time traffic rerouting is essential for addressing traffic congestion. Nonetheless, realizing such adaptability and customization is far from trivial.

Machine Learning (ML) techniques have emerged as promising enablers for on-demand service deployments, offering predictive and optimization capabilities. For example, Autoregressive Integrated Moving Average (ARIMA) and Deep Reinforcement Learning (DRL) have been leveraged to orchestrate on-demand deployment across multiple edge clouds [2]. Despite these advances, existing ML approaches suffer from the following limitations: *1) their effectiveness degrades when the model is not carefully defined and trained in a specific environment, an unavoidable reality in real-world vehicular systems; 2) their black-box nature hinders transparency and explainability, complicating real-time decision interpretation, 3) they lack the deep environmental understanding required to synthesize and interpret multimodal data in complex, rapidly changing contexts.*

Large Language Models (LLMs) have garnered substantial attention in the field of vehicular networks due to their greater knowledge representational power and ability to process enormous amounts of data, capturing patterns and associations for more accurate results [3]. More broadly, the concept of Large Telecom Models (LTMs) has been introduced as a paradigm for tailoring large-scale AI capabilities to meet the demands of the telecom ecosystem, spanning resource allocation, network optimization, and intelligent service provisioning across cloud-edge architectures [4]. Furthermore, Multimodal LLMs (MLLMs) extend these capabilities by integrating and interpreting multimodal data, including textual data and visual cues, and have been leveraged to improve decision-making in autonomous driving. However, the enormous size of LLMs/MLLMs makes it challenging for their seamless integration into the existing vehicular network architecture. Cloud-based LLMs can accommodate large-scale data and execute complex tasks with sufficient resources, yet they incur unacceptable latency and computation costs. Alternatively, edge-based LLMs offer reduced latency with acceptable computation cost margins; however, they are constrained by limited computation capacity and may only favor resource-efficient tasks.

Existing frameworks have explored cloud-edge collaboration as a viable approach to strategically position LLMs and Small LMs (SLMs) in resource-constrained environments, each performing a specific task based on task complexity and resource efficiency [5], [6]. For instance, some frameworks adopt a *"teacher-student paradigm"*, wherein a larger LM in the cloud generates guidance prompts, and a smaller LM finalizes responses at the edge [5]. Others leverage MLLM-based cloud-edge collaboration to improve environmental understanding and decision-making in the Internet of Things (IoT) networks [6]. However, these existing frameworks are customized for guidance prompts and inference decision-making, rendering them less adaptable for on-demand vehicular service deployment scenarios. Additionally, they lack memory augmentation, which limits their ability to retain contextual knowledge for future decision-making,

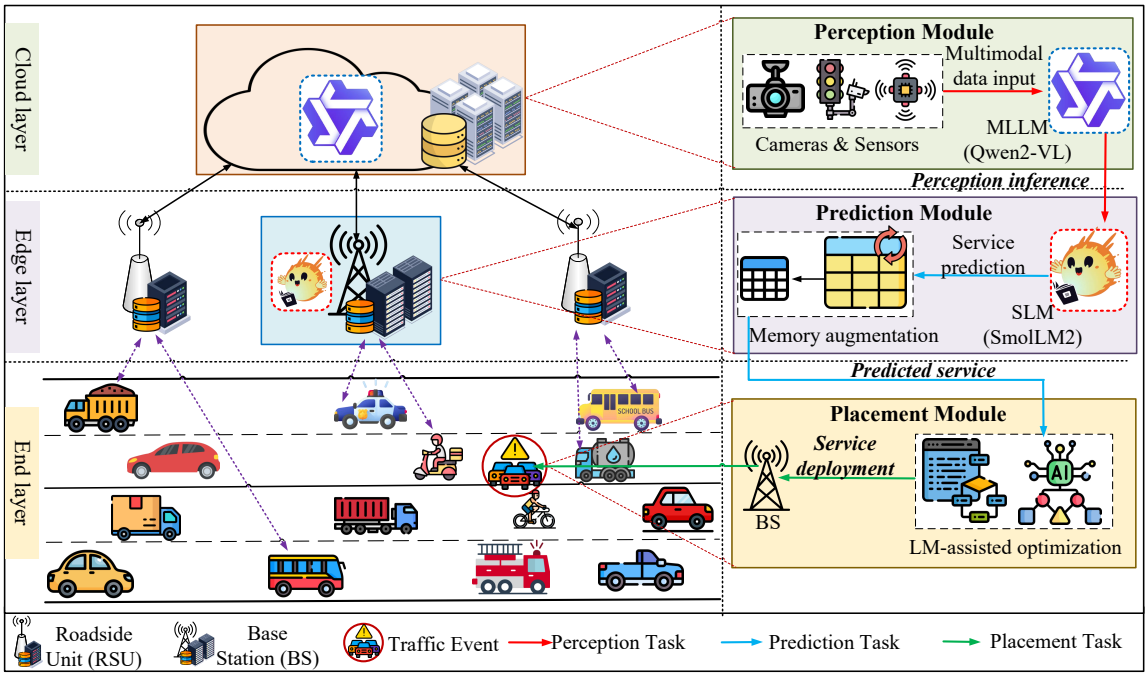

Figure 1: The L2S-LM architecture.

a phenomenon known as *"catastrophic forgetting"* [7]. Recently, Retrieval-Augmented Generation (RAG) and dynamically evolving knowledge databases have been identified as essential mechanisms for equipping large telecom models with up-to-date domain-specific knowledge and mitigating knowledge degradation in cloud-edge deployment settings [4]. However, their integration into on-demand vehicular service deployment pipelines remains largely unexplored.

Unlike existing frameworks, this article presents **L2S-LM**, a novel layered and collaborative LLM-SLM architecture tailored for on-demand vehicular service deployment in 6G networks. The proposed architecture hinges on three modular components: **Perception**, **Prediction**, and **Placement**. Specifically, a cloud-deployed MLLM (fine-tuned Qwen2-VL model) executes the perception module by analyzing raw multimodal vehicular datasets to extract high-level semantic representations and generate perception inferences. These inferences are then transmitted to the edge layer, where a lightweight SLM (fine-tuned SmolLM2 model), augmented with a memory mechanism, executes the service prediction module. This module predicts the most contextually appropriate service(s) required to address real-time traffic events based on the perception inferences. The prediction output is then stored in memory, allowing the model to retain contextual knowledge for future decision-making and mitigating catastrophic forgetting. Finally, the prediction results serve as input to an LM-assisted optimization algorithm, which executes the placement module to determine the optimal node among On-Board Unit (OBU) clusters and Roadside Units (RSUs) and allocate resources for service deployment. Experimental results based on a practical case study confirm the feasibility and effectiveness of the proposed architecture. The main novelty of our proposed architecture lies in its customization for on-demand vehicular service deployment and the memory augmentation mechanism that enables edge caching to avoid cloud offloading.

Overview of the L2S-LM Architecture

L2S-LM Architecture

The L2S-LM architecture spans the cloud, edge, and end layers of the vehicular network, as illustrated in Figure 1. We describe the components and functions of each layer in the architectural design.

End layer: The end layer consists of end users (such as vehicles) connected to infrastructure (such as RSUs) at the edge layer. Each vehicle is equipped with onboard cameras and other sensors that continuously collect raw multimodal data from the surrounding vehicular environment. The end layer serves as the source of multimodal data streams (visual data, textual data) generated by vehicles and hosts services once placement decisions are made. The vehicles transmit the raw multimodal data to higher layers for processing, semantic interpretation, and service prediction.

Edge layer: The edge layer is equipped with RSUs and Base stations (BSs) with moderate computational and storage capabilities. At its core is the SmolLM2 model [8], a fine-tuned SLM deployed alongside a memory augmentation mechanism. SmolLM2 integrates high-level semantic inferences from the cloud layer with historical contextual knowledge from the augmented memory (if any) to execute the prediction module. The prediction task is resource-efficient as it suggests with accuracy which specific service(s) are best suited to address real-time vehicular traffic events. The memory augmentation mechanism enables more consistent and informed predictions across recurring scenarios for future perception tasks.

Cloud layer: The cloud layer is equipped with high-performance computing and storage capabilities, and is responsible for handling resource-intensive perception tasks. At the cloud layer, a fine-tuned Qwen2-VL MLLM [9] is deployed to execute the perception module, where the MLLM processes raw multimodal data streams from the end layer to extract semantically rich representations of the vehicular environment. These representations capture salient features that are passed down to the SmolLM2 model at the edge layer to execute service prediction.

It is noteworthy that on-demand vehicular service deployment task execution at the end tier is assisted by the collaborative MLLM-SLM perception-prediction interactions in the cloud-edge layers. Rather than replicating the resource-intensive MLLM at the edge, we deploy an SLM since the MLLM output is unimodal text.

Operational Workflow

A typical operational workflow of the proposed L2S-LM architecture is described as follows:

The perception module is executed at the cloud layer:

1) The vehicles collect raw multimodal data (e.g., visual data from onboard cameras, textual data from other sensors) and transmit the data to the cloud via RSUs and BSs using uplink Vehicle-to-Infrastructure (V2I) communications.

2) The cloud-based MLLM (fine-tuned Qwen2-VL model) processes the raw multimodal data to perceive the environment by extracting high-level semantic features of the vehicular context (e.g., type of traffic incident, its severity, and the most probable cause), referred to as *perception inference*.

3) The resulting perception inference (structured as a unimodal textual output) is passed down to the edge layer to assist in downstream vehicular service prediction via Infrastructure-to-Infrastructure (I2I) links.

The prediction module is executed at the edge layer:

4) Upon receiving the perception inference from the cloud layer, a lightweight edge-based SLM (fine-tuned SmolLM2 model) interprets the inference and predicts the most appropriate service(s) required to handle the detected event (e.g., emergency response service for accidents, real-time traffic rerouting for congestion).

5) A memory augmentation mechanism stores the perception inference and service prediction outcome, which may be utilized to refine and improve similar future predictions through contextual recall. This is supported by Vehicle-to-Vehicle (V2V) sidelinks and edge connectivity.

6) A replica of the service prediction output is transmitted to the end layer for vehicular service placement optimization.

The placement module is executed at the edge or end layer:

7) An optimization algorithm (e.g., conventional or learning-based) leverages the prediction output to determine the best service placement option between local vehicle nodes (OBU clusters) and edge RSUs and allocates the necessary resources based on specific constraints.

8) The service is then deployed and executed to address the vehicular traffic event.

The primary communication bottleneck in L2S-LM is the V2I uplink for offloading raw multimodal data to the cloud. Memory augmentation mitigates this by reusing historical context, allowing most decision cycles to exchange compact semantic messages, i.e., inferences and predictions, rather than continuous raw streams. To ensure data privacy and security, we adopt the IEEE 1609.2 standard across all communication interfaces. V2I and I2I links utilize Transport Layer Security (TLS) for authenticated, encrypted data transmission, while V2V messages employ digital signatures for integrity [10].

L2S-LM Architecture: Key Modules

The proposed architecture executes an end-to-end *Perception-Prediction-Placement* pipeline for effective on-demand vehicular service deployment. The output of the preceding module serves as the input to the module below, enabling an interleaved modular architecture.

Perception Module

In our network scenario, the perception module is responsible for extracting high-level semantic representations from raw multimodal data (e.g., visual and textual data) collected by the vehicles to report contextual inference outputs about vehicular traffic events. It comprises two steps: 1) multimodal data collection and fusion, and 2) MLLM training and fine-tuning, as depicted in Figure 2.

Multimodal data collection and fusion: The perception process begins with the collection of heterogeneous data streams from various sources, including visual data from vehicle onboard cameras and traffic light cameras, as well as textual and numerical data such as sensor logs, weather data, and traffic incident reports. To ensure temporal and spatial alignment, incoming input data streams are grouped based on submission intervals and geographical location coordinates. To support practical downstream reasoning, the system instruction prompt, temporally aligned visual inputs, and structured textual-numerical data are interleaved and concatenated into a modality-aware composite sequence.

MLLM training and fine-tuning: The proposed L2S-LM architecture employs the publicly available pre-trained Qwen2-VL as the base model for its MLLM. The Qwen2-VL model primarily consists of a vision encoder (a Vision Transformer (ViT)) to process visual data, an LLM (original Qwen2 LLM) to process textual data, and a merger that connects the vision encoder and the LLM for token alignment. After the fused multimodal data is passed to the MLLM, the visual inputs are first divided into patches and embedded into a latent representation space compatible with the ViT encoder. These patches are processed through the ViT layers to extract a hierarchy of visual tokens that capture both spatial and semantic features. Simultaneously, the textual inputs are tokenized using the Qwen2 tokenizer and embedded into their respective latent space. The merger then aligns the visual and textual token dimensions through a patch reduction and projection mechanism, ensuring compatibility with the LLM's input format. This is done by reducing and aligning the dimensionality of the visual tokens to match the LLM's expected embedding size. Both the visual and textual embeddings are then concatenated and augmented with the Multimodal Rotary Positional Embeddings

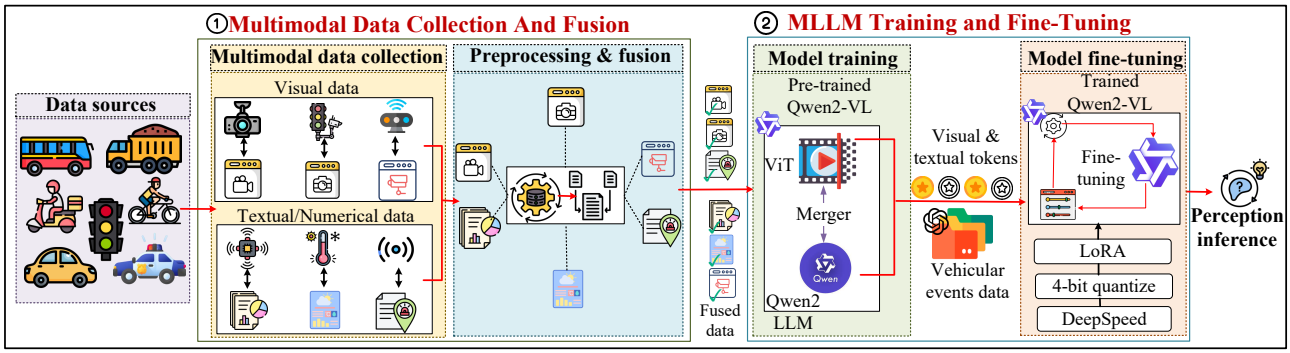

Figure 2: The perception module.

(M-RoPE) technique [9], which preserves modality-specific spatial and sequential structure during attention operations. This joint sequence is passed into the Qwen2-VL decoder blocks, where cross-modal attention mechanisms enable the model to reason over the entire input context and derive a semantically grounded perception of the vehicular scene.

Fine-tuning aligns the model output with the requirements of downstream modules by guiding it to extract, structured, task-relevant features such as pedestrian presence, road obstructions, scene visibility, and the root cause of congestion. Given the enormous size of the MLLM, directly fine-tuning full-precision parameters poses significant computational challenges. To address this, several optimization strategies can be employed as follows: i) Low-Rank Adaptation (LoRA) introduces trainable low-rank matrices into specific attention layers, significantly reducing the number of trainable parameters; ii) 4-bit quantization using the BitsAndBytes library compresses the trainable weights, minimizing memory overhead; iii) The DeepSpeed framework partitions model states, including optimizer and gradient buffers across CPU and GPU resources to optimize training efficiency. These fine-tuning optimization strategies enable the MLLM to be fine-tuned effectively for the perception task without compromising scalability. We fine-tune the language backbone using LoRA with rank $r = 16$ and scaling factor $\alpha = 32$, while keeping the vision encoder frozen. Training is conducted for up to 5 epochs with a micro-batch size of 4 samples/GPU and gradient accumulation to an effective batch size of 8, with early stopping based on validation loss to ensure stable convergence. We adopt the same LoRA configuration and training schedule for the prediction module. All configurations across the evaluated architectures are tuned using the same hyperparameter settings to ensure fair comparisons.

The final output of the perception module is a structured set of features that interpret the scene, which are serialized in a compact JSON format and tokenized before being forwarded as textual input to the prediction module at the edge layer.

Prediction Module

For on-demand vehicular service deployment, prediction involves suggesting actionable plans based on environmental perception and observation. The prediction module is responsible for interpreting the semantic perception inference and determining the most contextually appropriate on-demand service(s) to respond to detected vehicular traffic events. Operating at the edge layer, the prediction module consists of three stages: 1) perception-prediction fusion, 2) SLM reasoning and task adaptation, and 3) memory augmentation, as shown in Figure 3.

Perception-prediction fusion: The perception-prediction fusion component serves as the primary interface between the cloud-based perception module and the edge-based prediction module. It selects the perception inference generated by the MLLM (if no relevant prediction data is stored in memory) or relevant historical prediction data in memory (if any) for accurate service prediction. If a contextually similar event exists in memory, a top-k set of matching historical records is retrieved using similarity scoring based on semantic embeddings. These retrieved historical data, annotated with previously predicted service outcome, is used to construct an enriched representation, which serves as input to the SLM. However, if there is no similarity, only the perception inference is passed to the SLM for service prediction task execution.

SLM reasoning and task adaptation: The perception inference (or the historical data in memory, if there is a similarity) is passed to the edge-deployed SLM to generate new, deployable service predictions. The SLM is built on the SmolLM2 family of models, a suite of compact transformer-based models designed for resource-efficient natural language reasoning in resource-constrained environments such as local and edge devices. Unlike the cloud-based MLLM, which performs complex perception tasks, the SLM is optimized for resource-efficient prediction tasks. This separation of complexity ensures that the heavy computationally intensive perception process remains centralized in the cloud, while the lightweight task of SLM reasoning is performed at the edge.

To align the SLM with the service prediction objective, domain-specific fine-tuning is performed. This involves adapting the pre-trained SmoLM2 model to predict structured service predictions along with accompanying justifications. Such fine-tuning is performed by instruction-tuned prompting with adaptation data to condition the model's behavior. The instruction prompt prevents the SLM from hallucinating. The justifications support interpretability and allow the service placement optimizers to validate or override prediction results if necessary. This structure encourages the model to weigh the

context against available options and make plausible, context-sensitive predictions. Similar to the MLLM fine-tuning, the SmolLM2 fine-tuning incorporates parameter-efficient techniques such as LoRA, 4-bit quantization, and DeepSpeed to minimize training memory overhead.

Memory augmentation: To improve responsiveness and contextual generalization, the prediction module integrates a memory augmentation mechanism that functions as a retrieval-based caching system. This component stores past interactions, including semantic inferences, predicted services, and associated justifications, and retrieves contextually similar events to inform future predictions. By applying RAG principles [11], the system can recall and reuse relevant historical examples when new inputs resemble previously encountered events.

Each memory entry record is embedded into a shared vector space using lightweight language and visual encoders, then indexed in a vector database (e.g., FAISS or Qdrant) for efficient similarity search. When a new vehicular data input arrives, its embeddings are compared against the memory index using cosine similarity to retrieve a top-k set of the most similar cases. These examples are forwarded to the SLM to guide current service prediction. If highly similar matches are found, the system may bypass perception recomputation, significantly reducing end-to-end latency. Meanwhile, an asynchronous reranking and refinement process monitors the alignment between retrieved predictions and actual outcomes, updating the memory index over time to reflect evolving traffic patterns and improve future match quality. This feedback loop ensures that the memory component adapts and remains contextually accurate in dynamic vehicular environments.

The retrieval parameters are empirically set to $k = 3$ and a cosine-similarity threshold of $\tau = 0.85$. Additionally, refreshing the memory index every $N = 100$ queries reliably tracks data shifts while preserving index stability. Beyond offline fine-tuning, L2S-LM incorporates an online, drift-aware adaptation mechanism built on the memory submodule, where traffic events, their predicted services, and placement outcomes are stored. If performance deviates from operational targets, sampled memory data are used for efficient LoRA-based fine-tuning, after which the updated SLM adapter is swapped in while the backbone remains frozen.

Placement Module

In our vehicular service scenario, service placement marks the final phase of on-demand service deployment as it involves determining the most suitable node for hosting the predicted on-demand service(s) and allocating the necessary resources required to host such services. Within the L2S-LM architecture, the placement module operates at the end layer or edge layer and is responsible for translating the predicted service actions into concrete deployment decisions. The placement module executes a two-step task: 1) optimal node selection and 2) optimal resource allocation, as illustrated in Figure 4.

Optimal node selection: Node selection identifies the most appropriate execution option for hosting the predicted service recommended by the edge-deployed SLM. Service placement decisions consider factors such as network dynamics, candidate hosting nodes (RSUs and OBUs), their resource capacity,

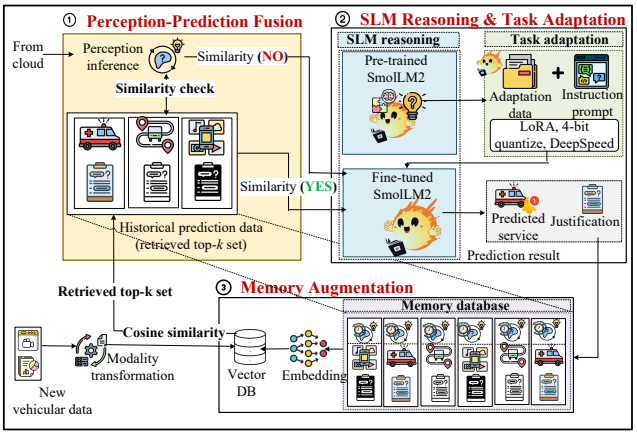

Figure 3: The prediction module.

proximity to the event, latency, availability, and nearby mobility patterns. RSUs offer high computational capacity and stable connectivity, making them ideal for service hosting since service predictions originate from the edge layer. However, their immobility may limit their responsiveness to spatially dynamic vehicular traffic events. Alternatively, nearby vehicular OBUs can serve as opportunistic hosting nodes, especially for localized events. While a single vehicular OBU may lack sufficient resource capacity, a cooperative OBU cluster can pool their resources to host services collaboratively [12]. If a particular node option satisfies the necessary conditions, it is selected as the optimal hosting node for service placement.

Optimal resource allocation: Once a candidate node is selected, the resource allocation step determines whether the node has sufficient computational and memory resources to support the service placement without compromising service quality. Resource allocation decisions are guided by Key Performance Indicators (KPIs) such as service satisfaction, resource utilization efficiency, and overall system utility. For instance, high bandwidth may be prioritized for video streaming services, while low-latency is critical for safety-critical applications.

The optimal node selection and resource allocation problem for on-demand vehicular service deployment can be formulated as a joint optimization problem. Traditional algorithms such as greedy heuristics and memetic algorithms [1] often fail to scale in dynamic environments and struggle with uncertain or incomplete system state information inherent in real-world vehicular networks. As a result, learning-based algorithms such as DRL [13], Multi-Agent RL (MARL), and Graph Neural Networks (GNNs) have emerged as promising alternatives, offering adaptability, scalability, and real-time decision-making capabilities in complex, evolving deployment scenarios. More importantly, the L2S-LM architecture enhances these methods by integrating LM-based context reasoning into the decision loop. By incorporating semantically rich perception inference from the cloud-deployed MLLM and the service prediction outcome from the edge-based SLM, the placement module can better anticipate deployment needs and prioritize hosting options aligned with real-time vehicular semantics.

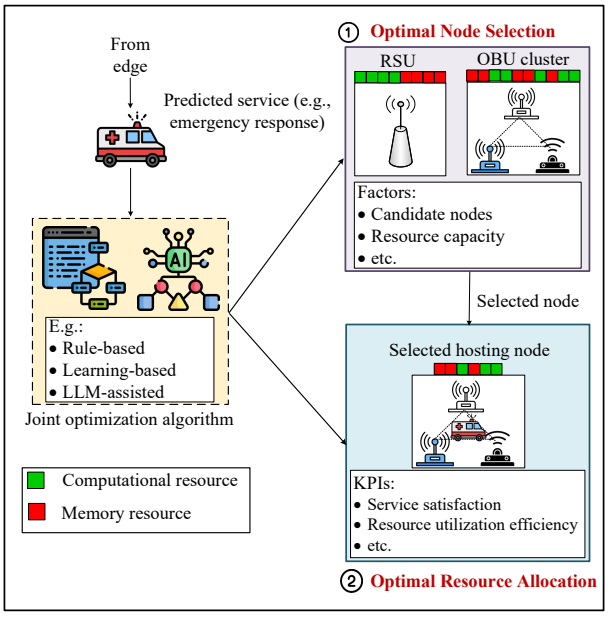

Figure 4: The placement module.

Specifically, an LM-assisted DRL optimization algorithm can be employed to place the services, since DRL agents enable fast online inference, making them suitable for time-sensitive placement decisions [3].

Case Study: Emergency Service Deployment

To validate the feasibility and effectiveness of the proposed L2S-LM architecture, we conduct a case study involving emergency service deployment in the event of an accident. We define the benchmark architectures and compare them with our proposed L2S-LM architecture as follows:

1) Traditional: This architecture does not involve any language models and serves as the fundamental baseline. In this setup, perception and prediction are implemented as a fused module at the edge nodes. Visual inputs are encoded using a ResNet-3D backbone, while textual modalities are converted to one-hot encodings, and numerical features are normalized via min-max scaling. The concatenated features are passed through a Multi-Layer Perceptron (MLP), which outputs a probability distribution over service candidates. The top-k services are then selected based on these probabilities.

2) Cloud-based: Both the perception and prediction modules are deployed in the cloud. The higher compute availability enables the use of large-scale models: `Qwen2-VL-7B` for perception and `Qwen2-7B` for prediction. This serves as the performance upper-bound baseline.

3) Hybrid (Memoryless L2S-LM): This architecture executes the computationally intensive perception task to the cloud (using `Qwen2-VL-7B`), and executes the resource-efficient prediction task at the edge (using lightweight `SmolLM2-135M` model).

4) L2S-LM (Proposed): Extending the Hybrid architecture, this setup integrates RAG mechanism that enables event-based reasoning via a memory database. Prior events are indexed and queried in real-time to accelerate inference and improve prediction plausibility.

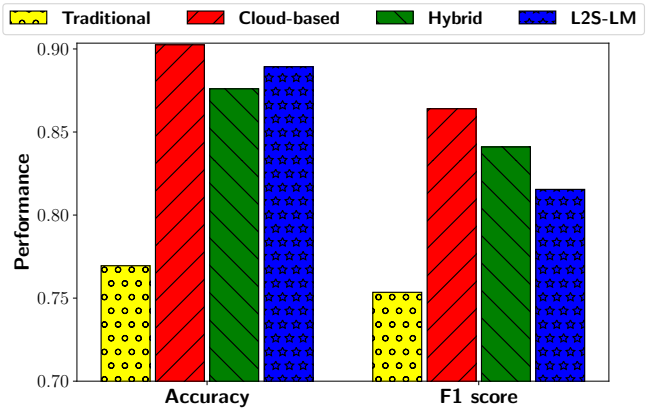

Figure 5: Performance on accuracy and F1 score.

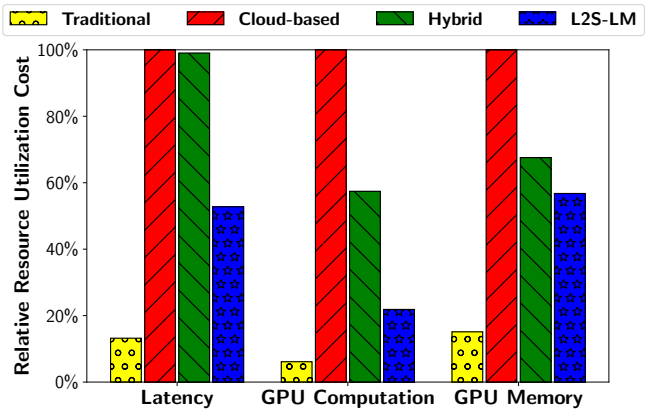

Figure 6: Relative resource utilization cost.

The multimodal datasets used in our experiments are constructed by integrating two publicly available data sources:

1) Highway Accident Detection and Classification Dataset [14]: It includes 2780 short video clips capturing various traffic incidents with textual metadata descriptions such as collisions, rollovers, and normal conditions.

2) Highway Traffic Videos Dataset [15]: It includes 262 video samples annotated with textual metadata of congestion levels, weather conditions, and timestamps.

We employ GPT-4o-mini to annotate each data sample with enriched contextual features and assign between 0 and 4 services, with an average of 1.88 services per sample. The total dataset comprises 3,042 samples split into 70% training, 10% validation, and 20% testing, with 30% of the textual modality removed from the test split to evaluate robustness to incomplete metadata. All performance evaluations are performed on the held-out test set using the fine-tuned models. All evaluations are conducted on the same hardware: NVIDIA Tesla V100 with 32 GB VRAM. For comparative analysis, all resource metrics are normalized relative to the cloud-based configuration, which serves as the 100% reference point.

Figure 5 compares the performance of our proposed L2S-LM architecture with the benchmark architectures in terms of

Accuracy and F1 score. Accuracy is calculated as the average of *strict match accuracy* and *soft match accuracy*, which rewards close predictions. F1 score is defined as the average of BERTScore F1 (for setups involving text generation, if applicable), *strict match* F1, and *soft match* F1. From Figure 5, we observe that the cloud-based setup achieves the highest performance, with accuracy and F1 score values of approximately 0.90 and 0.86, respectively. The Hybrid and L2S-LM architectures yield competitive results, while the traditional setup achieves the lowest performance. L2S-LM maintains strong performance in accuracy and F1 score of about 0.88 and 0.83, respectively, due to its memory-augmented reasoning at the edge. In contrast, the traditional architecture lacks language modeling entirely, relying on fixed feature extraction pipelines and shallow decision models, which limits its generalization and contextual adaptability.

Figure 6 compares the effectiveness of L2S-LM with the benchmarks in terms of latency, GPU computation consumption, and GPU memory consumption. Latency reflects the end-to-end time required for completing the perception and prediction tasks. From Figure 6, it is observed that the traditional architecture exhibits the lowest overall resource consumption, owing to its minimal parameter size and lack of advanced processing, albeit its low performance. The cloud-based architecture incurs the highest computational cost across all metrics, serving as the 100% reference point with 10.83 seconds latency, 93% GPU computation consumption, and 35.16 GiB GPU memory consumption. The hybrid model shows a notable reduction in GPU memory consumption and GPU computation consumption, offering a more efficient balance. Building upon this, the proposed L2S-LM architecture achieves even better efficiency by minimizing its dependence on cloud computation, reducing latency to 52.8% (5.72 seconds), GPU computation consumption to 21.9% (20.4%), and GPU memory consumption to 56.8% (19.97 GiB). Additionally, L2S-LM avoids redundant cloud-based perception in approximately 74% of cases with memory augmented predictions, efficiently reusing relevant historical context. To quantify the communication benefits, we consider $N = 100$ vehicles offload 5-second incident clips per minute at 720p encoding, yielding a baseline aggregate uplink demand of approximately 42 Mbps. With a memory reuse rate of 74%, the effective uplink reduces to approximately 11 Mbps, a fourfold reduction, confirming that L2S-LM maintains scalable bandwidth requirements as N increases. This demonstrates that edge caching with selective retrieval can offer substantial efficiency gains without significantly compromising service prediction performance.

To evaluate robustness to unseen traffic conditions, we conduct a 24-hour experiment under Normal, Rush-hour, and Bad-weather scenarios, with metrics aggregated hourly and averaged per scenario. Figure 7 shows the results of the generalization capability of L2S-LM compared with a static version (L2S-LM$_{static}$). When Rush-hour and Bad-weather are introduced, L2S-LM$_{static}$ decreases to Acc $\approx$ 0.80 and F1 $\approx$ 0.76. In contrast, L2S-LM uses this as a drift signal, and restores performance to Acc $\approx$ 0.85-0.88 and F1 $\approx$ 0.82-0.86, remaining within predefined operational thresholds.

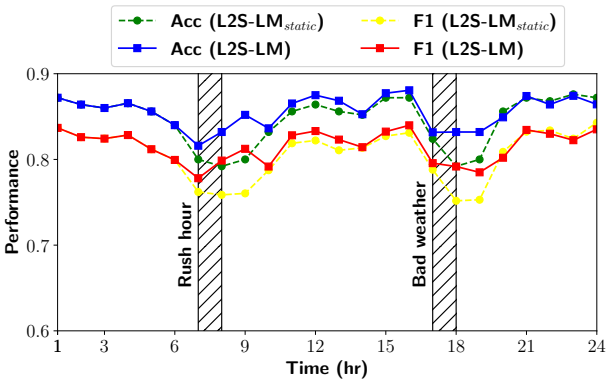

Figure 7: Generalization capability of L2S-LM.

In summary, we can conclude that our L2S-LM architecture achieves accuracy and F1 score performance comparable to that of the cloud-based architecture, while drastically reducing latency, GPU computation, memory consumption, and sustaining performance under distributional shifts. It achieves an effective trade-off between performance and responsiveness, achieving cloud-comparable accuracy and contextual precision.

Conclusion

In this article, we proposed L2S-LM, a collaborative architecture that combines LLM and SLM for on-demand vehicular service deployment in 6G networks. The system follows a modular design, with perception handled in the cloud, prediction at the edge, and service placement at the end layer. Specifically, a cloud-deployed MLLM perceives the vehicular environment and generates perception inferences. Then, an edge-deployed SLM processes the inferences to predict vehicular services suitable for addressing vehicular traffic events. By using the memory augmentation technique, the computation-intensive perception stage is avoided if similar historical predictions are stored in the memory database. Then, an LM-assisted optimization algorithm is employed to select the optimal nodes and allocate resources between RSUs and OBU clusters for service placement. Experimental results from a case study confirmed the efficacy of the proposed L2S-LM architecture in terms of acceptable accuracy and F1 score, as well as reduced latency and GPU resource consumption. Future work will explore adaptive LM-empowered multiple-node microservice deployment.

Acknowledgment

The authors acknowledge the contribution of Khalifa University for providing computing resources for this research. The corresponding author is Azzam Mourad.

Author Information

Amine Kidane Ghebreziabiher (amine.kidane@ku.ac.ae) is a Research Assistant with the KU 6G Research Center, Khalifa University, Abu Dhabi, 127788, UAE. He received his BSc. degree from Khalifa University in 2024. His current research interests include Applied AI, Agentic and Gen AI, Cyber Security, Network and Service Optimization targeting IoT and IoV, Cloud/Fog/Edge Computing, Vehicular and Mobile Networks, 5G/6G networks, vehicular networks, and reinforcement learning.

Gordon Owusu Boateng (Gordon.Boateng@xjtlu.edu.cn) is an Assistant Professor with Xi'an Jiaotong-Liverpool University (XJTLU), Suzhou, 215123, China. He received his Ph.D. degree from the University of Electronic Science and Technology of China in 2023. His current research interests include 5G/6G wireless networks, blockchain, reinforcement learning, vehicular networks, and large language models. He is a Member of IEEE.

Daniel Ayepah-Mensah (daniel.mensah@ku.ac.ae) is a Postdoctoral Fellow with the KU 6G Research Center, Khalifa University, Abu Dhabi, 127788, UAE. He received his Ph.D. degree from the University of Electronic Science and Technology of China in 2024. His current research interests include 5G wireless networks, artificial intelligence, network virtualization, and edge computing. He is a Member of IEEE.

Rabeb Mizouni (rabeb.mizouni@ku.ac.ae) is an Associate Professor with the Department of Electrical Engineering and Computer Science, Khalifa University, Abu Dhabi, 127788, UAE. She received her Ph.D. degree from Concordia University in 2007. Her current research interests include the deployment of context-aware mobile applications, crowd sensing, artificial intelligence, IoT, and blockchain.

Azzam Mourad (azzam.mourad@ku.ac.ae) is an AI Advisor to the President and Professor of Computer Science at Khalifa University and Founding Director of the Artificial Intelligence and Cyber Systems Research Center at the Lebanese American University. He received his Ph.D. from Concordia University, Canada (2008). His research interests include Applied AI, Agentic and Gen AI, Cyber Security, Federated Machine Learning, Network and Service Optimization targeting IoT, IoV, and Cloud/Fog/Edge Computing. He is an IEEE senior member.

Hadi Otrok (Hadi.Otrok@ku.ac.ae) is an Associate Professor with the Department of Electrical Engineering and Computer Science (EECS), Khalifa University, Abu Dhabi, 127788, UAE. He received his Ph.D. degree from Concordia University in 2008. His current research interests include the domain of blockchain, reinforcement learning, crowd sensing and sourcing, ad hoc networks, and cloud security. He is a Senior Member of IEEE.

Jamal Bentahar (jamal.bentahar@ku.ac.ae) is a Visiting Professor with Khalifa University, Abu Dhabi, 127788, UAE. He received his Ph.D. degree from Laval University in 2005. His current research interests include the areas of computational logics, reinforcement learning, multi-agent systems, service computing, game theory, and software engineering.

Sami Muhaidat (sami.muhaidat@ku.ac.ae) is a Professor and the Associate Dean for Research with the College of Computing and Mathematical Sciences, Khalifa University, Abu Dhabi, 127788, UAE. He received his Ph.D. degree from University of Waterloo in 2006. His current research interests include advanced digital signal processing techniques for wireless communications, intelligent surfaces, machine learning for communications, optical communications, and massive multiple-access techniques. He is a Senior Member of IEEE.

REFERENCES

[1] H. Sami, R. Saado, A. E. Saoudi, A. Mourad, H. Otrok, and J. Bentahar, "Opportunistic uav deployment for intelligent on-demand iov service management," *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, pp. 3428–3442, 2023.

[2] Y. Huang, B. Feng, Y. Cao, Z. Guo, M. Zhang, and B. Zheng, "Collaborative on-demand dynamic deployment via deep reinforcement learning for iov service in multi edge clouds," *Journal of Cloud Comp.*, vol. 12, no. 1, p. 119, 2023.

[3] G. O. Boateng, H. Sami, A. Alagha, H. Elmekki, A. Hammoud, R. Mizouni, A. Mourad, H. Otrok, J. Bentahar, S. Muhaidat, C. Talhi, Z. Dziong, and M. Guizani, "A survey on large language models for communication, network, and service management: Application insights, challenges, and future directions," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2025.

[4] A. Shahid and et al., "Large-scale ai in telecom: Charting the roadmap for innovation, scalability, and enhanced digital experiences," 2025. [Online]. Available: https://arxiv.org/abs/2503.04184

[5] Y. Yao, Z. Li, and H. Zhao, "Gkt: A novel guidance-based knowledge transfer framework for efficient cloud-edge collaboration llm deployment," 2024. [Online]. Available: https://arxiv.org/abs/2405.19635

[6] Y. Hu, D. Ye, J. Kang, M. Wu, and R. Yu, "A cloud-edge collaborative architecture for multimodal llms-based advanced driver assistance systems in iot networks," *IEEE Internet of Things Journal*, pp. 1–1, 2024.

[7] D. Zhu, Z. Sun, Z. Li, T. Shen, K. Yan, S. Ding, K. Kuang, and C. Wu, "Model tailor: Mitigating catastrophic forgetting in multi-modal large language models," 2024. [Online]. Available: https://arxiv.org/abs/2402.12048

[8] L. B. Allal and et al., "Smollm2: When smol goes big – data-centric training of a small language model," 2025. [Online]. Available: https://arxiv.org/abs/2502.02737

[9] P. Wang and et al., "Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution," 2024. [Online]. Available: https://arxiv.org/abs/2409.12191

[10] "Ieee standard for wireless access in vehicular environments–security services for applications and management messages," *IEEE Std 1609.2-2016 (Revision of IEEE Std 1609.2-2013)*, pp. 1–240, 2016.

[11] S. Liu, Z. Yu, F. Huang, Y. Bulbulia, A. Bergen, and M. Liut, "Can small language models with retrieval-augmented generation replace large language models when learning computer science?" in *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1*, ser. ITiCSE 2024. New York, NY, USA: Association for Computing Machinery, 2024, p. 388–393. [Online]. Available: https://doi.org/10.1145/3649217.3653554

[12] H. Sami, A. Mourad, and W. El-Hajj, "Vehicular-obus-as-on-demand-fogs: Resource and context aware deployment of containerized micro-services," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 778–790, 2020.

[13] M. Chahoud, H. Sami, A. Mourad, H. Otrok, J. Bentahar, and M. Guizani, "On-demand model and client deployment in federated learning with deep reinforcement learning," *IEEE Internet of Things Journal*, pp. 1–1, 2025.

[14] L. Kezebou, V. Oludare, K. Panetta, J. Intriligator, and S. Agaian, "Highway accident detection and classification from live traffic surveillance cameras: a comprehensive dataset and video action recognition benchmarking," in *Multimodal Image Exploitation and Learning 2022*, S. S. Agaian, V. K. Asari, S. P. DelMarco, and S. A. Jassim, Eds., vol. 12100, May 2022, p. 121000M.

[15] A. Chan and N. Vasconcelos, "Probabilistic kernels for the classification of auto-regressive visual processes," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, pp. 846–851 vol. 1.