Adaptive ML-Enabled Edge-Cloud System Framework for Safe and Efficient Autonomous Systems

Eunho Cho[0000−0002−4293−945X] and In-Young Ko[0000−0002−3843−263X]

Korea Advanced Institute of Science and Technology (KAIST)
Deajeon, Republic of Korea
{ehcho, iko}@kaist.ac.kr

Abstract. Machine Learning (ML)-enabled systems like Autonomous Driving Systems (ADSs) face challenges meeting safety and performance requirements in diverse environments, especially in resource-constrained, latency-sensitive edge-cloud settings. These challenges often arise from ML models' limitations, including poor generalization to unseen conditions. Adaptive algorithms using ML system switching have been proposed, but existing approaches frequently lack generalizability, support for common black-box systems, and effective use of distributed edge-cloud resources. This paper presents a novel adaptive ML-enabled Edge-Cloud system framework to address these shortcomings. Our framework combines cloud-based pre-runtime analysis, which leverages simulation for behavioral understanding and scenario-to-system mapping, with collaborative edge-cloud runtime adaptation featuring dynamic ML model switching. It supports black-box systems and aims to balance safety and efficiency by utilizing appropriate edge and cloud resources situationally. Preliminary CARLA-based evaluation of the edge runtime component suggests our framework can potentially improve the safety-efficiency trade-off compared to single-model ADSs in some scenarios. This work offers insights for designing adaptive edge-cloud systems and identifies future directions, including robust cloud analysis and effective edge-cloud collaboration. Findings suggest this edge-cloud approach can advance the feasibility and reliability of adaptive ML systems for real-world autonomous applications.

Keywords: ML-Enabled Systems · Autonomous Driving Systems · Edge-Cloud Computing · Adaptive Systems · Simulation-Based Testing

1 Introduction

Autonomous systems, particularly Autonomous Driving Systems (ADSs), are increasingly integrated within the broader edge-cloud computing ecosystem. Modern ADSs generate substantial sensor data at the edge, selectively offload data to the cloud for large-scale analysis and model refinement, and receive updated models back at the edge. This collaborative edge-cloud architecture aims for safer roads, reduced congestion, and more efficient mobility services [14].

ADSs rely on machine learning (ML) components for perception, decision-making, and control tasks. However, these ML components face challenges due to the unpredictable nature of real-world scenarios—including diverse traffic patterns, unexpected behaviors, and varying environmental conditions [17, 16]. Ensuring reliable performance, especially under edge constraints (resource limits, latency requirements), remains an open challenge.

Frameworks like Autoware [11] and Apollo [1] use modular designs with independently optimized components. Despite such architectures, achieving consistent safety and efficiency across diverse driving scenarios with a fixed set of ML models is difficult. ML models risk being too large for practical computational budgets or too small and overfitting specific scenarios [9]. Static ML systems, therefore, often struggle with adaptability, particularly when facing scenarios unseen during training, highlighting the need for dynamic solutions in edge-cloud environments.

Adaptive approaches using runtime ML system switching have been proposed to mitigate these challenges [13, 6]. However, most existing approaches assume white-box accessibility, requiring predictable model behaviors and limiting practical applicability. This limits practical applicability because many state-of-the-art ML models used in ADSs, particularly deep neural networks, function as complex black boxes, making their internal states difficult to predict or analyze directly for adaptation purposes. Moreover, these methods often don't fully leverage both cloud computational strengths and edge real-time capabilities simultaneously. Thus, frameworks that seamlessly integrate edge-cloud collaboration for effectively handling black-box ML systems under varying conditions are critically needed.

To address the limitations of static models in diverse/unseen scenarios and the applicability constraints of existing white-box adaptive approaches, particularly concerning black-box systems and effective edge-cloud collaboration, we present an adaptive ML-enabled edge-cloud system framework designed to overcome these limitations. Our approach combines a cloud-driven pre-runtime phase (leveraging extensive simulations for behavioral analysis and scenario mapping) with a collaborative runtime phase where edge and cloud systems jointly identify scenarios and dynamically select optimal ML systems. This two-phase strategy maximizes cloud resources for exhaustive pre-runtime analyses and edge speed for real-time adaptability. Our prototype implementation using the CARLA simulation platform [7] integrates both a high-performance deep-learning agent and a computationally efficient rule-based agent, showing promising preliminary results regarding safety-efficiency trade-offs.

The main contributions of this study are:

– **Edge-cloud adaptive framework**: Formalized an edge-cloud adaptive framework supporting black-box ML systems, defining cloud roles (pre-runtime analysis) and edge-cloud collaboration roles (runtime adaptation).
– **Empirical validation**: Quantified safety-efficiency trade-offs of adaptive ML switching via preliminary CARLA-based validation.

– **Insights and future research directions**: Provided insights and future directions for scenario generation, knowledge base maintenance, and deployment in practical edge-cloud environments.

The remainder of this paper is structured as follows: Section 2 reviews relevant literature. Section 3 introduces our proposed adaptive framework. Section 4 describes the experimental setup and presents preliminary results. Section 5 discusses implications, limitations, and future research. Finally, Section 6 summarizes our findings.

2 Related Work

The adaptive system framework, employing system or model switching, is critical for ensuring ML system adaptability in dynamic environments. This technique dynamically transitions between models/systems to maintain optimal performance, safety, and quality under varying circumstances.

Several studies explore adaptive frameworks. For instance, researchers have proposed strategies like predictive control with reconfigurable models [4], model selection using transfer reinforcement learning (RL) in Open IoT [15], anomaly-aware adaptation via RL for cyber-physical systems [6], and QoS-based switching to manage performance uncertainties [13].

However, these valuable studies often have limitations. Many frameworks are constrained to specific environments or predefined scenarios, limiting generalizability. Some depend on accessible system behaviors, making them unsuitable for black-box systems. It is very difficult or impossible to determine in which situation the system behavior will react as desired for many high-performance ML models used in practice. Furthermore, existing works frequently focus on specific aspects like efficiency or anomaly detection, often neglecting the complex interplay with other critical factors such as overall system safety.

Recent research explores Edge AI optimization considering resource constraints [20] and novel cloud-edge collaborative architectures [10]. Yet, these often do not directly tackle the specific challenge of dynamically adapting black-box ML models for safety-critical autonomous driving systems. Our work aims to bridge this gap by proposing a framework integrating adaptive black-box ML switching within a structured Edge-Cloud approach tailored for ADS safety and efficiency.

3 Adaptive ML-Enabled Edge-Cloud System Framework

Our framework utilizes an Edge-Cloud architecture comprising two main phases: a pre-runtime phase executed on cloud resources and a runtime phase operating across edge and cloud infrastructure. This section first discusses the overall approach, followed by detailed examinations of the pre-runtime and runtime phases.

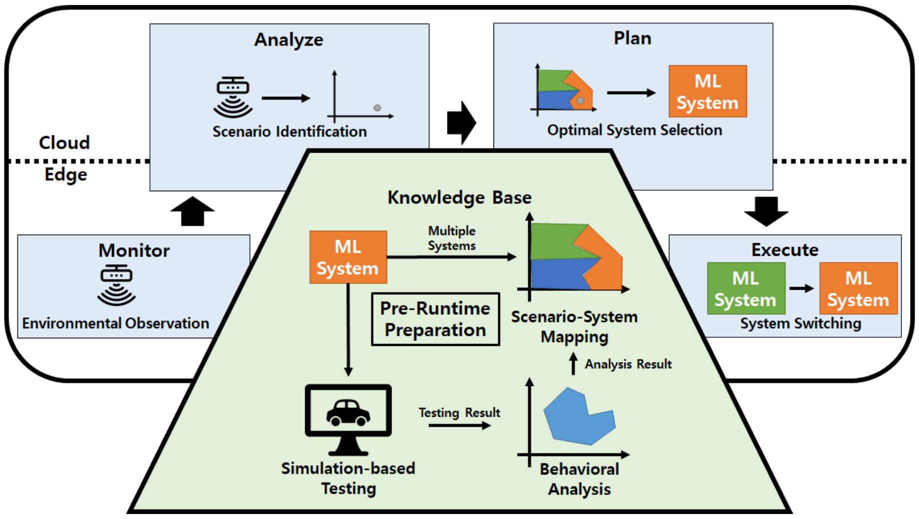

Fig. 1. Overall Adaptive ML-Enabled Edge-Cloud System Framework

Figure 1 illustrates the framework's flow, divided into two phases. The lower section depicts the cloud-based pre-runtime phase, which involves simulation-based testing, behavioral analysis, and scenario-system mapping to prepare the system knowledge base. This phase evaluates candidate ML systems through extensive simulation, analyzes their behavior to identify optimal operating subspaces, and maps the most suitable system to each subspace within the knowledge base, leveraging cloud computational power.

The upper section shows the runtime phase, based on the distributed MAPE-K loop [12], a standard model for self-adaptive systems. The edge monitors the environment (Monitor); edge and cloud collaboratively identify the current scenario subspace (Analyze); they collaboratively select the optimal ML system using the knowledge base (Plan); and the edge executes the system switch (Execute). This combination of cloud pre-runtime preparation and distributed runtime adaptation enables stable and effective adaptation in complex environments.

In this framework, a 'Scenario' represents the specific operational context, often captured as a vector encompassing environmental factors such as weather and road type, system goals like waypoints, and current system state variables including speed and position. We assume multiple ML systems exist, each optimized for different scenarios or data distributions. The primary goal of the pre-runtime phase is to analyze system behavior across diverse scenarios via large-scale simulations, identify the scenarios best suited for each system's operation, and systematically compile this information into a knowledge base that informs runtime decision-making.

3.1 Pre-Runtime Phase

Executed on the cloud due to computational demands, the pre-runtime phase includes three key steps: simulation-based testing, behavioral analysis, and scenario-system mapping.

Simulation-based Testing First, we evaluate ML system performance using extensive simulations (e.g., using CARLA) across diverse environments. These scenarios, encompassing complexities like varying weather, road networks, and traffic in ADS contexts, enable the collection of performance data for each ML system. Scenario generation prioritizes diversity to cover a broad spectrum of operating conditions, grounding the identification of suitable operating scenarios for each system.

Simulation-based testing is crucial for evaluating ML systems, especially when real-world testing is risky or infeasible. It allows systematic performance evaluation under diverse, controlled conditions. Platforms like CARLA [21] offer realistic physics-based simulations suitable for analyzing safety and efficiency metrics in autonomous systems like Autonomous Driving Systems (ADSs). The scale and computational demands of such simulations are well-suited for cloud execution. Simulation testing characterizes ML system performance. Prior work includes surrogate model-based frameworks [8] and search-based approaches to identify hazard boundaries [19], demonstrating simulation's ability to define operational limits. In our framework, simulation testing is a key component of the cloud-based pre-runtime phase.

Behavioral Analysis Next, each ML system's behavior is analyzed using the simulation data, focusing on safety and efficiency. For ADSs, safety metrics might include detailed collision types, involving pedestrians or vehicles for example, off-road excursions, route completion ratio, and compliance with traffic regulations such as traffic signal adherence and stop sign compliance. Efficiency metrics cover computational cost, like average inference time per frame or peak memory usage, and network resource consumption required by each ML system. This analysis assesses whether systems meet safety requirements and satisfy QoS criteria within different scenario subspaces.

Scenario-System Mapping This mapping step is crucial because different ML systems exhibit varying performance trade-offs, notably between safety and efficiency, under different scenarios. Based on the behavioral analysis, this step systematically assigns the best-performing system to each identified scenario subspace according to pre-defined objectives, for instance prioritizing safety over efficiency, creating a reliable decision guide known as the knowledge base for the runtime phase. This involves cloud-based multi-objective optimization, potentially using evolutionary algorithms or rule-based heuristics. The resulting mapping is stored in the knowledge base, possibly as a compact decision tree or a hash map for efficient runtime lookup, and potentially cached at the edge for runtime access, directly enabling the system's adaptive capabilities.

3.2 Runtime Phase

The runtime phase operates using the distributed MAPE-K (Monitor, Analyze, Plan, Execute, Knowledge) loop across edge and cloud resources. We assume the edge device is equipped with necessary sensors. Each step leverages edge and cloud strengths to maintain safety and efficiency in complex environments.

Environmental Observation The runtime process begins with observing the environment in real-time using onboard edge sensors (e.g., cameras, LiDAR, radar). This collected data forms the input for the subsequent analysis step.

Scenario Identification Following observation, the current scenario subspace is determined using the observed edge data. A scenario involves initial environment/system states and operational goals. Observations identify the current state and goals, but unobservable variables may lead to multiple possible scenarios, thus forming a scenario subspace. This analysis is performed collaboratively on edge and cloud. Rapid analysis of immediate sensor data is performed by the edge device for local context identification, potentially using lightweight convolutional neural networks to identify environment, or other cars. The cloud leverages larger datasets or more powerful models for deeper analysis, perhaps employing Long Short-Term Memory networks for predicting traffic or accessing aggregated complex mobility data, providing broader context or predictions. The result is an identified scenario subspace incorporating both edge and cloud perspectives.

Optimal System Selection Based on the identified scenario subspace, the most suitable ML system is selected by referencing the knowledge base. This planning step also occurs collaboratively on edge and cloud. A quick selection is made by the edge using the identified local context and the potentially cached knowledge base to address immediate needs. A more sophisticated selection or refinement is performed by the cloud, considering broader goals or complex trade-offs. This decision can potentially update the edge's initial plan, for instance, based on predicted traffic congestion patterns or system-wide energy optimization goals. The final decision on the optimal ML system combines edge and cloud inputs, aiming for optimal safety and efficiency based on both immediate needs and longer-term objectives.

System Switching Finally, the execution step involves switching to the selected ML system on the edge device. This transition must occur in real-time with minimal disruption. The execution requires edge technologies capable of minimizing transition delays and ensuring system stability post-transition. Furthermore, the performance of the newly activated ML system should ideally be monitored, with results potentially reported to the cloud to refine the knowledge base for future decision-making improvement.

4 Investigation

This section presents a preliminary evaluation of the runtime adaptation component within our proposed adaptive ML-enabled edge-cloud framework. A simple prototype adaptive ADS assesses practical applicability, focusing on safety and efficiency. As discussed in the Introduction, achieving both safety and efficiency consistently across diverse driving scenarios is a primary challenge for ADSs, especially considering edge resource constraints and the limitations of static or existing adaptive approaches. Our proposed framework aims to improve this balance through adaptive ML switching in an edge-cloud context. Therefore, this preliminary evaluation focuses on quantifying the potential benefits regarding these two critical aspects, guided by the following research questions (RQs):

RQ1 (Safety): What advantages does the adaptive ADS offer in terms of safety compared to conventional ADS, either holistically or in specific scenarios?

RQ2 (Efficiency): What advantages does the adaptive ADS provide in terms of efficiency compared to conventional ADS?

Experiments ran on Ubuntu 22.04 (Intel Xeon 4215R, 3x RTX A5000 GPUs 24GB, 128GB RAM). The replication kit containing the source code used in this study is publicly available at [3].

4.1 Experiment Design

We implemented a simple adaptive ADS prototype focusing on runtime adaptation at the simulated edge. The prototype uses CARLA [7], our simulated edge environment, and Leaderboard benchmarks [2] to evaluate safety and efficiency across various scenarios. We compared the adaptive mechanism against single ML systems regarding safety and efficiency at the edge.

CARLA [7] is a widely used open-source ADS simulator providing diverse maps and dynamic components. We used CARLA datasets, originally from Transfuser [5] and InterFuser [18] training, for testing. Key test scenarios included 'longest6', '42routes', 'town05_short/long', 'town06_long', and 'town10_short'. The CARLA Leaderboard [2] executes these scenarios under predefined conditions using modules like scenario runner.

Evaluation focused on safety and efficiency. Safety metrics from the CARLA Leaderboard included: *Route Score*, representing the percentage of the route completed; *Penalty Score*, reflecting points deducted for incidents where 1 is perfect; and *Composed Score*, providing an overall safety measure. We also introduced *Scenario Dominance Count*, measuring the number of scenarios where an agent achieved the highest composed score. Efficiency was measured by the decision time ratio, calculated as CARLA simulation time divided by real-world time.

The prototype combines Transfuser [5] and the NPC Agent sourced from the CARLA Leaderboard, simulating edge runtime adaptation. Transfuser is a complex DNN agent utilizing camera and LiDAR fusion via self-attention; it is capable of handling challenging scenarios but computationally heavy for continuous edge execution. The NPC Agent is a simple, computationally efficient rule-based baseline suitable for edge resources but limited in complex situations. The prototype activates Transfuser near intersections, defined as within 50 meters, and uses the NPC Agent otherwise, simulating edge-based scenario identification and system selection. This logic prioritizes safety with Transfuser in higher-risk intersections and efficiency with NPC on simpler road segments, balancing the trade-offs within the edge environment.

The prototype employs a simple rule-based knowledge base, simulating the output of the envisioned cloud pre-runtime analysis. The underlying principle is that Transfuser offers higher safety at greater computational cost, while NPC is efficient but less safe. The mapping reflects this: Transfuser is selected for high-risk intersections prioritizing safety, and NPC for other roads prioritizing efficiency. This allows the prototype to dynamically adapt at the edge, balancing safety and efficiency based on simple scenario detection.

4.2 Experiment Result

We evaluated the adaptive ADS prototype against the NPC and Transfuser ADS to address RQ1 (Safety) and RQ2 (Efficiency).

Table 1. Safety and Efficiency Metrics for ADS Systems

Scenarios	NPC					Transfuser					Adaptive				
	Route	Penalty	Composed	Dominance	Time	Route	Penalty	Composed	Dominance	Time	Route	Penalty	Composed	Dominance	Time
longest6	39.8	0.237	5.4	1	0.157	42.7	0.799	24.4	14	0.094	61.7	0.419	24.4	21	0.110
42routes	75.1	0.315	21.7	0	0.078	85.5	0.901	79.3	23	0.074	69.4	0.490	37.6	1	0.074
town05_short	79.6	0.375	27.9	2	0.129	97.0	0.877	85.5	29	0.101	93.4	0.856	80.8	1	0.104
town05_long	63.2	0.136	5.1	0	0.133	100.0	0.459	45.9	9	0.098	92.5	0.330	30.2	1	0.119
town06_long	66.0	0.088	1.6	0	0.174	94.5	0.613	56.8	9	0.103	92.5	0.177	16.6	1	0.106
town10_short	31.2	0.406	18.0	0	0.098	100.0	0.810	81.0	8	0.069	83.7	0.653	58.9	1	0.074

Regarding RQ1 (Safety), Table 1 and Figure 2 present the safety and efficiency results. Transfuser consistently achieved the highest composed scores, while NPC performed poorly, highlighting the inherent trade-off between performance and computational simplicity relevant to edge suitability. Although the adaptive ADS often scored below the pure Transfuser, it demonstrated robustness. Notably, it matched Transfuser's composed score in the 'longest6' scenario and achieved the highest score, indicating dominance, in 21 scenarios overall according to Table 1. This result suggests that, concerning RQ1, the potential effectiveness of edge-based adaptation. However, lower scores in specific scenarios, such as 'town06_long,' indicate limitations of the prototype's simple adaptation rule, pointing to the need for refinement through the full edge-cloud framework involving, for example, more sophisticated scenario identification or a richer knowledge base.

In relation to RQ2 (Efficiency), Figure 3 illustrates computational efficiency via time ratios. Transfuser's consistently low time ratio signifies high computational cost, potentially problematic for resource-constrained edge devices. Conversely, NPC was the most efficient but demonstrated poor safety performance. The adaptive ADS achieved intermediate efficiency. This result indicates that,

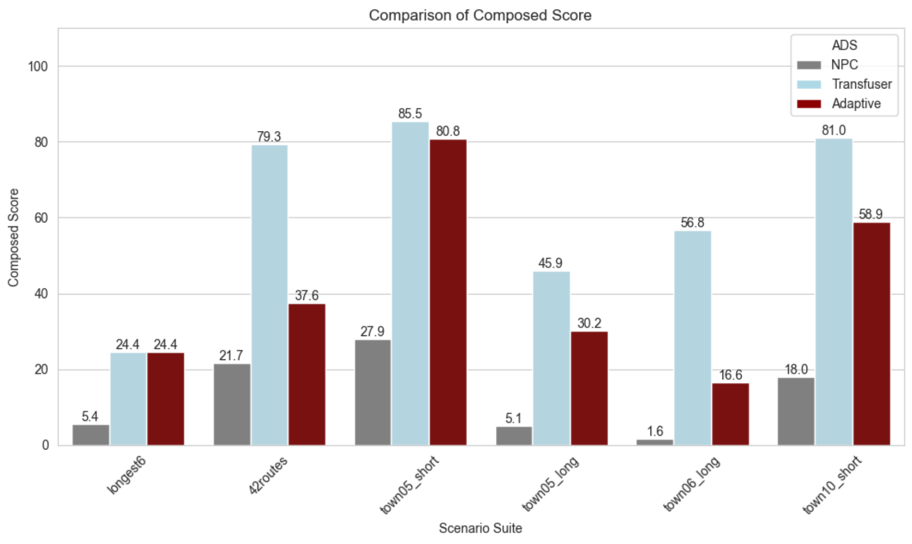

Fig. 2. Comparison of Composed Score for ADS Systems

concerning RQ2, the adaptive approach successfully balanced safety needs with edge resource usage by selectively engaging the computationally heavier Transfuser only when necessary, thus demonstrating a clear efficiency advantage over running Transfuser continuously.

In summary, addressing RQ1 and RQ2, these preliminary results suggest that edge-based adaptation, as demonstrated by the prototype, can offer safety performance comparable to the best-performing single model (Transfuser) in specific scenarios while achieving significantly greater computational efficiency suitable for edge deployment. This indicates the potential of the edge adaptation component within our broader edge-cloud framework to address the core challenge of balancing safety and efficiency. However, the observed limitations underscore the need for future work focusing on the complete framework implementation, including cloud-based pre-runtime analysis and more sophisticated edge-cloud coordination mechanisms to improve the adaptation strategy and achieve robust performance across a wider range of scenarios.

5 Discussion

5.1 Investigation Analysis

Our preliminary evaluation indicates the framework's edge runtime adaptation component can potentially balance safety and efficiency across diverse scenarios. The adaptive prototype outperformed monolithic systems by dominance count in some scenarios (Table 1) and matched Transfuser's safety score in 'longest6' while

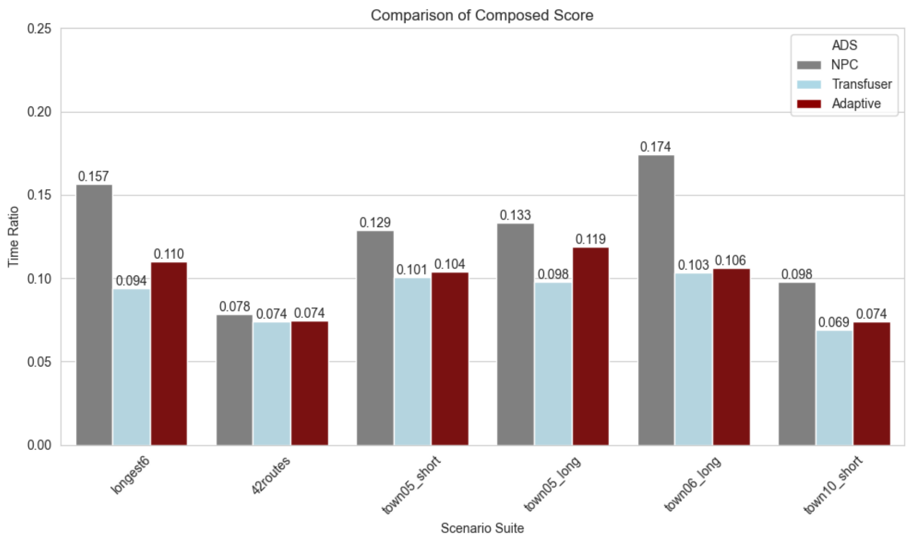

Fig. 3. Comparison of Time Ratio for ADS Systems

being more computationally efficient (Figure 3). This suggests dynamic edge switching, guided by scenario perception, is a promising approach for adapting ADS performance under varying edge conditions.

However, the experiments also revealed limitations, emphasizing the importance of cloud-based pre-runtime analysis and careful component selection for edge execution. The NPC Agent's struggles with unexpected events (e.g., pedestrians, obstacles) negatively impacted safety when it was selected by the simple edge adaptation logic in non-intersection scenarios. This issue stems from the NPC's design as a simple rule-based algorithm, highlighting the challenge of relying solely on simple edge models for complex, unforeseen events and underscoring the need for robust cloud-based pre-analysis to identify such weaknesses or provide mechanisms for escalating to more capable models or cloud intervention.

Similarly, Transfuser's occasional safety failures in certain intersection scenarios, despite high overall scores, demonstrate that the adaptive system's performance is ultimately bounded by the capabilities and potential flaws of the underlying models deployed at the edge. While edge switching can optimize system selection based on known characteristics derived from pre-analysis, it inherits the drawbacks of its constituent systems if the cloud pre-runtime analysis fails to adequately characterize these limitations or if the edge cannot reliably detect the critical conditions requiring a specific model.

Consequently, these observed limitations underscore the critical role of the cloud-based pre-runtime phase, involving extensive simulation-based testing and behavioral analysis, in building a comprehensive and accurate knowledge base.

This knowledge base is essential for informing effective edge runtime adaptation logic, enabling accurate mapping of scenarios to the most suitable edge-executable systems, and understanding the operational boundaries and limitations of each component. Therefore, while this study validates the potential of the edge-based adaptive approach, it simultaneously highlights the critical need for thorough cloud-based pre-analysis as an indispensable element for optimizing both safety and efficiency in the design of adaptive edge-cloud ADS systems.

5.2 Future Directions

To overcome the limitations identified and fully realize the framework's potential, future work should first address robust scenario generation techniques for the cloud pre-runtime analysis. Simulation scenarios are critical for evaluating the performance and identifying the limitations of candidate ADS models intended for edge deployment. Cloud-based scenario generation methods should systematically explore the operational boundaries of these models by designing diverse scenarios including edge cases, such as unexpected obstacles or highly complex intersections. These methods need to balance criticality (exploring key boundary conditions) and variety (covering diverse environments), while enabling automation to reduce the time and cost associated with building the pre-runtime knowledge base.

Second, mapping scenarios to optimal system behaviors and identifying the scenario subspaces where specific edge models excel or fail remain significant challenges, primarily addressed within the cloud component during pre-runtime analysis. Addressing these requires advanced techniques. Exploration-based techniques can automatically identify critical scenarios within cloud simulations, efficiently pinpointing model weaknesses. Reinforcement learning is well-suited for training adaptive switching policies governing edge behavior, allowing the edge component to learn and respond effectively based on cloud-derived models or policies.

Furthermore, effective collaboration mechanisms between the edge and the cloud require in-depth research. Defining clear criteria for when scenario analysis should be handled solely by the edge versus requiring cloud interaction is a key challenge in designing the collaboration logic. This includes designing efficient communication protocols, maintaining data consistency, managing potential conflicts between edge and cloud decisions, and developing strategies for dynamic task allocation and resource management considering real-time requirements, communication constraints, and computational loads at both edge and cloud.

Future research will prioritize integrating comprehensive cloud-based behavioral analysis with sophisticated self-adaptive techniques executed at the edge, ensuring effective edge-cloud collaboration. Such efforts aim to maximize the safety and efficiency of autonomous driving systems by leveraging the strengths of both edge and cloud resources, while ensuring reliable performance in complex environments. Ultimately, advancing edge-cloud adaptive frameworks is expected to improve the feasibility of autonomous driving technologies for commercialization and play a critical role in developing safe and efficient autonomous systems.

6 Conclusion

We proposed an adaptive ML-enabled Edge-Cloud framework to enhance autonomous system safety and efficiency. It combines cloud pre-runtime analysis with collaborative edge-cloud runtime adaptation (including ML switching) to handle diverse requirements. Preliminary CARLA evaluation validated the edge adaptation component's effectiveness. The experimental results demonstrated that the proposed framework, through edge adaptation potentially improved trade-off between safety and efficiency compared to conventional single ML system approaches in certain scenarios.

A simplified prototype confirmed the edge adaptation's applicability and highlighted future work: improving test scenario generation for cloud analysis, developing robust scenario-system mapping (e.g., using search or RL), and refining edge-cloud collaboration. Implementing and evaluating the complete framework is the crucial next step.

Acknowledgments. This work was partly supported by the Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT)(RS-2025-02218761, 50%), IITP grant funded by the Korea government(MSIT)(RS-2025-00406245, 25%), Samsung Electronics(25%).

References

1. Baidu apollo team (2017), apollo: Open source autonomous driving. https://github.com/ApolloAuto/apollo, accessed: 2024-12-09
2. Carla autonomous driving leaderboard. https://leaderboard.carla.org/, accessed: 2024-12-09
3. Replication kit. https://github.com/EunhoCho/AdaptiveADS/, accessed: 2025-04-17
4. Amir, M., Vahid, F., Givargis, T.: Switching predictive control using reconfigurable state-based model. ACM transactions on Design Automation of Electronic systems (tODAEs) **24**(1), 1–21 (2018). https://doi.org/10.1145/3267126
5. Chitta, K., Prakash, A., Jaeger, B., Yu, Z., Renz, K., Geiger, A.: Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**(11), 12878–12895 (2023). https://doi.org/10.1109/TPAMI.2022.3200245
6. Cho, E., Yeo, G., Jee, E., Bae, D.H.: Anomaly-aware adaptation approach for self-adaptive cyber-physical system of systems using reinforcement learning. In: 2022 17th Annual System of Systems Engineering Conference (SOSE). pp. 7–12. IEEE (2022). https://doi.org/10.1109/SOSE55472.2022.9812671
7. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: Carla: An open urban driving simulator. In: Conference on robot learning. pp. 1–16. PMLR (2017)
8. Haq, F.U., Shin, D., Briand, L.: Efficient online testing for dnn-enabled systems using surrogate-assisted and many-objective optimization. In: Proceedings of the 44th international conference on software engineering. pp. 811–822. IEEE (2022). https://doi.org/10.1145/3510003.3510188

9. Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M.M.A., Yang, Y., Zhou, Y.: Deep learning scaling is predictable, empirically. arXiv preprint arXiv:1712.00409 (2017). https://doi.org/10.48550/arXiv.1712.00409
10. Ji, X., Gong, F., Wang, N., Xu, J., Yan, X.: Cloud-edge collaborative service architecture with large-tiny models based on deep reinforcement learning. IEEE Transactions on Cloud Computing (2025)
11. Kato, S., Tokunaga, S., Maruyama, Y., Maeda, S., Hirabayashi, M., Kitsukawa, Y., Monrroy, A., Ando, T., Fujii, Y., Azumi, T.: Autoware on board: Enabling autonomous vehicles with embedded systems. In: 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS). pp. 287–296. IEEE (2018). https://doi.org/10.1109/ICCPS.2018.00035
12. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer **36**(1), 41–50 (2003). https://doi.org/10.1109/MC.2003.1160055
13. Kulkarni, S., Marda, A., Vaidhyanathan, K.: Towards self-adaptive machine learning-enabled systems through qos-aware model switching. In: 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE). pp. 1721–1725. IEEE (2023). https://doi.org/10.1109/ASE56229.2023.00172
14. Muhammad, K., Ullah, A., Lloret, J., Del Ser, J., de Albuquerque, V.H.C.: Deep learning for safe autonomous driving: Current challenges and future directions. IEEE Transactions on Intelligent Transportation Systems **22**(7), 4316–4336 (2020). https://doi.org/10.1109/TITS.2020.3032227
15. Noguchi, H., Isoda, T., Arai, S.: Shared trained models selection and management for transfer reinforcement learning in open iot. In: 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC). pp. 2170–2176. IEEE (2021). https://doi.org/10.1109/SMC52423.2021.9658890
16. Serban, A.C.: Designing safety critical software systems to manage inherent uncertainty. In: 2019 IEEE International Conference on Software Architecture Companion (ICSA-C). pp. 246–249. IEEE (2019). https://doi.org/10.1109/ICSA-C.2019.00051
17. Shafaei, S., Kugele, S., Osman, M.H., Knoll, A.: Uncertainty in machine learning: A safety perspective on autonomous driving. In: Computer Safety, Reliability, and Security: SAFECOMP 2018 Workshops, ASSURE, DECSoS, SASSUR, STRIVE, and WAISE, Västerås, Sweden, September 18, 2018, Proceedings 37. pp. 458–464. Springer (2018). https://doi.org/10.1007/978-3-319-99229-7_39
18. Shao, H., Wang, L., Chen, R., Li, H., Liu, Y.: Safety-enhanced autonomous driving using interpretable sensor fusion transformer. In: Proceedings of The 6th Conference on Robot Learning. vol. 205, pp. 726–737. PMLR (2023)
19. Sharifi, S., Shin, D., Briand, L.C., Aschbacher, N.: Identifying the hazard boundary of ml-enabled autonomous systems using cooperative coevolutionary search. IEEE Transactions on Software Engineering **49**(12), 5120–5138 (2023). https://doi.org/10.1109/TSE.2023.3327575
20. Surianarayanan, C., Lawrence, J.J., Chelliah, P.R., Prakash, E., Hewage, C.: A survey on optimization techniques for edge artificial intelligence (ai). Sensors **23**(3), 1279 (2023)
21. Zhang, J.M., Harman, M., Ma, L., Liu, Y.: Machine learning testing: Survey, landscapes and horizons. IEEE Transactions on Software Engineering **48**(1), 1–36 (2020). https://doi.org/10.1109/TSE.2019.2962027