BenchCouncil Transactions on Benchmarks, Standards and Evaluations 2 (2022) 100086

Contents lists available at ScienceDirect

BenchCouncil Transactions on Benchmarks, Standards and Evaluations

journal homepage: www.keaipublishing.com/en/journals/benchcouncil-transactions-on-benchmarks-standards-and-evaluations/

Research article

Edge AIBench 2.0: A scalable autonomous vehicle benchmark for IoT–Edge–Cloud systems

Tianshu Hao [a,b,*], Wanling Gao [a], Chuanxin Lan [a], Fei Tang [a,b], Zihan Jiang [a,b], Jianfeng Zhan [a,b]

[a] Research Center for Advanced Computer Systems, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[b] University of Chinese Academy of Sciences, Beijing, China

ARTICLE INFO

Keywords:
IoT–Edge–Cloud
Benchmark
Autonomous vehicles
Scalable

ABSTRACT

Many emerging IoT–Edge–Cloud computing systems are not yet implemented or are too confidential to share the code or even tricky to replicate its execution environment, and hence their benchmarking is very challenging. This paper uses autonomous vehicles as a typical scenario to build the first benchmark for IoT–Edge–Cloud systems. We propose a set of distilling rules for replicating autonomous vehicle scenarios to extract critical tasks with intertwined interactions. The essential system-level and component-level characteristics are captured while the system complexity is reduced significantly so that users can quickly evaluate and pinpoint the system and component bottlenecks. Also, we implement a scalable architecture through which users can assess the systems with different sizes of workloads.

We conduct several experiments to measure the performance. After testing two thousand autonomous vehicle task requests, we identify the bottleneck modules in autonomous vehicle scenarios and analyze their hotspot functions. The experiment results show that the lane-keeping task is the slowest execution module, with a tail latency of 77.49 ms for the 99th percentile latency. We hope this scenario benchmark will be helpful for Autonomous Vehicles and even IoT–edge–Cloud research. Now the open-source code is available from the official website https://www.benchcouncil.org/scenariobench/edgeaibench.html.

1. Introduction

As a typical complex real-world application, IoT–Edge–Cloud systems consist of "a diversity of AI and non-AI modules with huge code sizes and long and complicated execution paths" [1]. Moreover, many emerging IoT–Edge–Cloud computing systems are yet implemented or are too confidential to reveal their technical details, not to mention sharing the source code. For example, a typical IoT–Edge–Cloud system – autonomous vehicles – runs 100 million lines of code in just one car [2]. Overall, they are too tricky or costly to replicate the code or even their execution environments; hence, their benchmarking is very challenging.

Even if we can replicate the application completely, directly using the application as the benchmark have several pitfalls. Real-world applications often have many instantiation biases. That is to say, real-world applications or systems are trapped in limited design and implementation points in a high-dimension space [3]. Previous work [4] has discussed the root cause of the instantiation bias. A workload is hierarchically implemented in a modern computer system: a problem definition, an algorithm, an intermediate representation, an ISA-specific representation, and a micro-architectural representation. From top to down, the design and implementation spaces increase explosively. However, for maintaining user experience or saving the software and hardware ecosystem investment, users adhere to existing products, tools, platforms, and services, which the previous work called technology inertia [3,5]. The technology inertia traps the real-world solution to a problem into a specific exploration path — a subspace or even a point at a high-dimension solution space. While profiling has been applied to various aspects of benchmarking complex real-world applications [1], the profiling technique helps little in overcoming this limitation.

Gao et al. [1] proposed a scenario benchmarking methodology to attack the above challenge. They proposed several rules to distill a real-world application scenario from a high-level requirement specification into a combination of essential AI and non-AI tasks as a scenario benchmark. Meanwhile, They identify primary modules in the critical paths of a real-world scenario from the system implementation level as they consume the most system resources and are the core focuses for system design and optimization. However, they fail to consider the complex IoT–Edge–Cloud scenarios. This paper extends the scenario benchmarking methodology for IoT–Edge–Cloud systems. For the first time, Hao et al. [6] propose an end-to-end view in benchmarking IoT–Edge–Cloud systems, considering all three layers: client-side devices,

* Corresponding author at: Research Center for Advanced Computer Systems, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China.
E-mail addresses: haotianshu@ict.ac.cn (T. Hao), gaowanling@ict.ac.cn (W. Gao), lanchuanxin@ict.ac.cn (C. Lan), tangfei@ict.ac.cn (F. Tang), jiangzihan@ict.ac.cn (Z. Jiang), zhanjianfeng@ict.ac.cn (J. Zhan).

https://doi.org/10.1016/j.tbench.2023.100086
Received 20 November 2022; Received in revised form 13 February 2023; Accepted 13 February 2023
Available online 16 February 2023

edge computing layer, and cloud servers. But their methodology has flaws. For example, they fail to consider the problem definition and instantiation bias. Moreover, they only implemented several component benchmarks in isolation without realistic interactions, which cannot constitute an end-to-end view. In addition, their workloads are not scalable.

The autonomous vehicles case is selected by most state-of-the-art benchmarks as a representative scenario and has the typical features of IoT–Edge–Clod systems [7–11]. Moreover, autonomous vehicles may be the most safety-critical scenario because it is crucial to human life. Therefore, building a unified, reasonable, and general benchmark set for autonomous vehicles is essential. There are several benchmarks for autonomous driving, such as KITTI [12], CAVBench [11], and Chauffeur [13]. However, they lack the scenario view to construct the benchmark, which will lead to a lack of the performance of the whole system. Therefore, in this paper, we choose autonomous vehicles as the case study to create a scalable scenario benchmark for IoT–Edge–Cloud systems, benefiting users to evaluate and improve their systems and applications.

The autonomous vehicles scenario is highly complex. Various AI vision workloads and critical decision tasks are presented in an autonomous car. These numerous tasks generate substantial input data, and these premises bring uncertainty to system function [14]. The Society of Automotive Engineers (SAE) classifies the quality of automation of a system into six levels of autonomous driving systems; the higher the level, the higher the system's performance, with L5 representing full automation [15]. While today's most advanced autonomous driving systems rarely reach L4 and L5, industrial companies are more focused on developing L2 and L3 level technologies, and there are still specific bottlenecks in the current level [16]. In summary, it is crucial to establish a unified, reasonable, and general benchmark set for autonomous driving, which will benefit the research and development of systems and applications in the field of autonomous vehicles.

In this paper, based on the state-of-the-art benchmarking methodology [1,3,4], we select autonomous vehicles as a research case to establish a scalable scenario-l benchmark for IoT–Edge–Cloud systems, which reduces the complexity of the system while maintaining the typical characteristics and critical execution path. This benchmark facilitates users in evaluating the system's performance and improving the algorithm. Finally, we conduct experiments using this IoT–Edge–Cloud scenario benchmark to analyze the critical task workloads in autonomous driving.

We sum up our main contributions as follows:

1. In order to ensure that the system's features are preserved as much as possible during the distilling process, we propose six distilling rules to simplify the scenario based on the characteristics of autonomous vehicles.
2. We propose the first scenario benchmark for the IoT–Edge–Cloud systems and provide the reference implementation. In addition, we also implement a scalable framework to support different sizes of workloads.
3. In the experiment section, we test two thousand autonomous vehicle tasks the end devices sent and measure the tail latency of each module. The results show the slowest execution module is the lane-keeping task and the convolution operations are the hotspot functions. Therefore, a scenario-based benchmark will help users find the bottleneck module of a system.

We organize the rest of this paper as follows. Section 2 summarizes the complexity of the autonomous vehicle scenario, the problem definition and instantiating in benchmark construction, and related work. Section 3 introduces the construction of scenario-level benchmarks. Section 4 introduces our scalable edge computing architecture. Section 5 performs evaluation. Section 6 concludes.

2. Problem definition and solution instantiation

Zhan [3] points out that a benchmark needs three processes: problem definition, solution instantiation, and measurement. We follow this guide to build our benchmark. Due to the different IoT, Edge, Cloud systems, workloads, and performance requirements, defining and instantiating the problems and their solutions is challenging.

2.1. Problem definition

Initially, this paper focuses on the problem of how to help users to get better performance from an IoT–Edge–Cloud system. Thus we extract a few critical workloads and provide a scalable benchmark to evaluate the systems to meet the performance requirements.

Secondly, we take autonomous vehicles, the most representative IoT–Edge–Cloud scenario, as the case study in this paper. Like most IoT–Edge–Cloud scenarios, autonomous vehicles have many complexity and entanglement among different components of the architecture and workloads. A comprehensive autonomous vehicle system may include many processing tasks. The driving automation is taken into six levels according to the international standard SAE J3016 with reference to the performance of the dynamic driving task (DDT) on a sustained basis [15]. Therefore, we take the DDT applications as the primary concern to instantiate the problem.

Thirdly, we extract the representative workloads and formalize them with a directed acyclic graph-based (DAG) model. Then we distill their critical path to build a scenario benchmark by the scenario benchmarking methodology [1]. We design and provide a workload reference implementation that reflects the characteristics of real scenarios.

To meet different users' requirements of the scales, we design and implement a scalable benchmark based on the scenario benchmark framework—scalable architecture helps the system allocate resources and workloads.

2.2. Complexity of autonomous vehicles scenario

Like most IoT–Edge–Cloud scenarios, an autonomous vehicle system has numerous application-level components. These components carry out a lot of communication across three layers of IoT–Edge–Cloud system architectures, making the system more complicated. The distributed three-layer architecture needs computing resources scaling and workload allocation of multiple layers. Accordingly, the scalability of the benchmark is also important to adapt to different sizes of workloads and meet the performance requirements of different users. However, unlike other scenarios, an autonomous vehicle system has its own characteristics. We summarize them below.

1. **The system sophistication .** The entire autonomous vehicle system involves a wide range of communications and data interaction. Meanwhile, it is also filled with numerous perception, planning, decision-making, and other autonomous driving tasks. The entire system processes massive volumes of data while running those intricate AI and non-AI algorithms in real time. Different design strategies provide difficulties for both hardware and software systems.
2. **Varied environmental factors.** During the driving process, the car will encounter various natural weather conditions (e.g., fog and snow) and complex terrain factors (e.g., mountains and hills), which will impact sensor data collection and the accuracy of AI tasks like objection recognition. Additionally, the existing autonomous driving system may not have an accurate judgment in extreme weather [17]. Hence, a reliable autonomous vehicle system must take into account a variety of weather conditions.

3. **Massive amount of input data.** Autonomous cars are equipped with many sensors, GPS positioning modules, and cameras for data collection, which will generate a large amount of heterogeneous input data [18] constantly. Moreover, multiple onboard cameras will keep collecting information about the surrounding environment. The system needs to consider how and where this data is processed, stored, and trained.
4. **The high demand for accuracy.** Automated driving tasks require absolutely correct decisions from the autonomous driving system. However, many tasks in the present AI models cannot achieve accuracy above 90% [19]. Additionally, there will be more uncertainties in the autonomous driving environment, such as sudden braking of the vehicle in front, pedestrians entering the road, and other unexpected situations. Consequently, safety can also be achieved during stable driving.
5. **Stable network performance.** As a typical IoT–Edge–Cloud system, an autonomous vehicle system requires real-time data interaction with cloud data centers and edge servers throughout the entire vehicle network. To support this, a high-bandwidth and high-performance network environment is required.
6. **Limited computing resources.** Real-time task processing has high requirements for computer resources due to the enormous amount of data. However, the processing capability of in-vehicle chips is constrained. Therefore, it needs to develop a lightweight model for these AI tasks to match the in-vehicle computing system is a significant issue. Numerous improved AI model pruning techniques [20,21] are now being presented to overcome the obstacle and meet the real-time requirement.
7. **High energy consumption.** Autonomous vehicles are equipped with numerous sensors and powerful processing chips, which have high energy consumption. According to studies, the overall power consumption of cars will rise by 2.8 to 4 percentage points to enable self-driving capabilities [22]. With the development of 5G technology, the energy demands of network communication will increase.

As summarized above, real autonomous driving systems are pretty complicated, making it difficult to fully and accurately model these characteristics in creating a representative benchmark.

2.3. Related work

In recent years, the field of autonomous vehicles with AI technologies has started to acquire traction. There is some relevant benchmarking research work for autonomous driving.

KITTI [12] is a vision benchmark suite for autonomous driving. It proposes stereo and optical vision data collected from the camera and the laser scanner. However, its purpose is to evaluate the vision algorithms' performance, not the whole autonomous driving system.

CAVBench [11] is the first benchmark suite for edge computing systems. It summarizes four scenarios and implements six AI workloads for autonomous vehicles. It takes an end-to-end view considering edge computing architecture. Nevertheless, it lacks a whole scenario-level view.

Chauffeur [13] is an open-source benchmark for autonomous driving. It implements end-to-end pipelines considering sensing, planning, and actuation processes. But it also did not consider the whole picture of the autonomous vehicle based on end-edge-cloud three-layer architecture.

In conclusion, the state-of-the-art autonomous vehicle benchmarks lack the scenario-level view to consider the whole scenario picture. They concentrate on specific algorithms, AI workloads, or hardware performance. However, an autonomous vehicle scenario is typical in IoT–Edge–Cloud systems, which must consider the components and the whole system's performance. Therefore, we need to distill the key module of the system and create a new scenario benchmark to simulate the real-world system's performance.

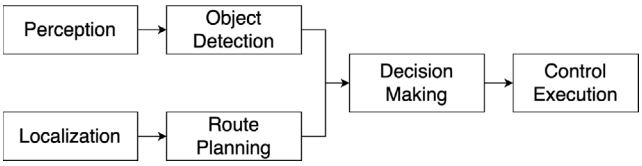

Fig. 1. Task flow chart of autonomous vehicles.

3. Creating the scenario benchmark

Based on the above challenges and motivations, we present the methodology and construction process for building an autonomous vehicle scenario-level benchmark for an intelligent edge computing system.

3.1. Specifying the autonomous vehicle scenario

An autonomous driving system mainly consists of a three-layer processing structure of perception, decision planning, and control execution module. The perception and control layers can be secured by configuring multi-layer redundant hardware systems. Thus, in the current research on autonomous driving systems, we focus mainly on the core algorithms for decision planning. The main emphasis in creating scenario benchmarks is likewise on decision planning-related artificial intelligence task modules.

According to the grading table of the international standard SAE J3016, it classifies driving automation into six levels with reference to the automation of dynamic driving tasks (DDT), DDT fallback, and object and event detection and response (OEDR) tasks on continuous driving systems. OEDR is a subtask of the DDT, which includes real-time object identification, classification, and other AI tasks. When a dynamic driving task fails, the system must perform the DDT fallback [15]. As a result, we instantiate our benchmark problem with the dynamic driving task as the primary concern. As dynamic driving is the fundamental task of autonomous driving, according to which we classify typical dynamic driving tasks and summarize the scenarios of autonomous vehicles in IoT–Edge–Cloud systems.

As shown in Fig. 1, the workflows of a complete set of dynamic tasks for autonomous driving include perception, location, path planning, object detection, and final decision-making. The perception module collects data through sensors and cameras, the localization module combines GPS module and map information to locate the vehicle's position, and the route planning module carries out a path planning task to determine an appropriate driving route according to the user's destination. The recognition task contains the recognition of vehicles, roads, pedestrians, obstacles, and traffic sign lights [23]. Finally, based on these parallel tasks, the vehicle-centric processor makes judgments regarding the current situation and decides to control the vehicle physically.

An autonomous vehicle currently has an intelligent edge chip with deep learning model processing capability, which can handle common lightweight AI autonomous driving tasks in real-time. However, it still needs to collaborate with cloud data centers and edge servers to execute tasks during the vehicle driving process better. In summary, autonomous vehicles use three layers of IoT–Edge–Cloud system resources to carry out diverse tasks.

As shown in Fig. 2, we used a set of directed acyclic graph (DAG) models to formalize the overall autonomous vehicle's tasks.

Large computing tasks or tasks with low real-time requirements are usually offloaded to the cloud data center for execution. At the same time, the cloud data centers also execute the task of offline training and ongoing retraining of the model. In the Internet of Vehicles, the cloud data center must communicate with all vehicles and make the whole vehicle network scheduling decisions.

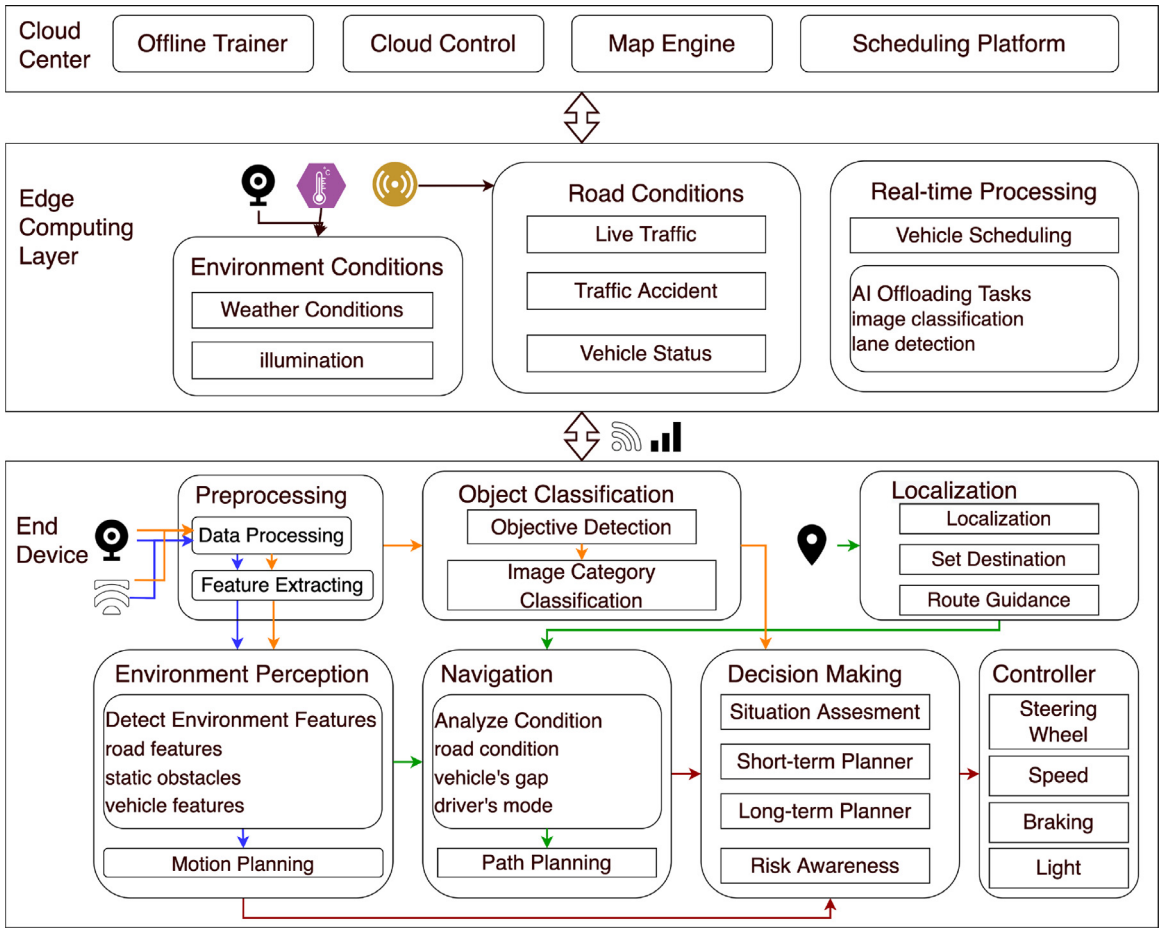

Fig. 2. The DAG model of autonomous vehicles in an IoT–Edge–Cloud system.

Autonomous vehicles connect to edge servers nearby when they move. These edge servers gather roadside environmental data, road information, and near-end vehicle data in real time, data that the vehicle's sensors often cannot collect because of blind spots and other issues. And the edge server will deliver it to nearby vehicles in a local area network. At the same time, with the guarantee of communication, autonomous driving vehicles will send tasks that cannot be processed in real-time by the onboard chip to the edge servers with sufficient computing power for processing. An excellent way to deal with issues like heterogeneous computing and energy consumption in autonomous vehicles is to offload jobs to the edge computing layer.

The smart chip on the vehicle side handles the primary autonomous driving workflow. The route planning and navigation tasks are executed by the vehicle in accordance with the user's instructions and GPS location data. In this procedure, the vehicle's sensors and cameras will gather data in real-time and pre-process them at the vehicle's end so that it can constantly recognize the environment, conduct perception tasks, and detect objects. The vehicle will simultaneously receive data from the edge server and cloud data center for integration. Finally, the vehicle's decision-making module will make decisions based on the information feedback from different modules and finally send the control commands.

3.2. Distilling rules for autonomous vehicles scenario

From Fig. 2, it is clear that formalizing the whole IoT–Edge–Cloud scenario is very complex. If the scenario benchmark is implemented accordingly, it will generate hundreds of millions of lines of code [24] and a vast amount of data, which is not conducive to users evaluating the system. Therefore, this section simplifies the autonomous vehicle scenario to extract several interdependent execution modules. Our work is inspired by the previous work [1] on the distilling rules for complex scenarios. And hence, the distilled modules can perform the critical tasks of an autonomous driving system while retaining the complexity and challenge of the system.

First, we propose a set of distilling rules for autonomous driving tasks based on real-world experience with autonomous driving, with reference to the industry's autonomous driving benchmark [11,12].

1. Retain only representative tasks among those that make use of similar models and serve similar purposes.
   In the process of autonomous driving, there are various types of object recognition and detection tasks, which include obstacle recognition, pedestrian recognition, traffic signal recognition, route recognition, etc. Most activities also share similar processing logic and critical path and are typically completed in two steps: detection and classification, with the exception of road route recognition in lane keeping. First, the object's location needs to be detected and localized in the video image, and then classification is performed to complete the recognition of the object. Therefore, we extract the critical traffic signal recognition from these tasks to ensure driving safety and the lane detection task to ensure vehicles obey traffic rules.
2. Prune the tasks executed on the cloud and edge servers and those in parallel with the user-end tasks.
   In IoT–Edge–Cloud systems, the user-end devices, edge servers, and servers in the cloud data center execute tasks in parallel and do not affect each other. Therefore, the training and scheduling tasks on the cloud do not affect the vehicle driving process. As a result, we prune this part of the tasks. At the same time,

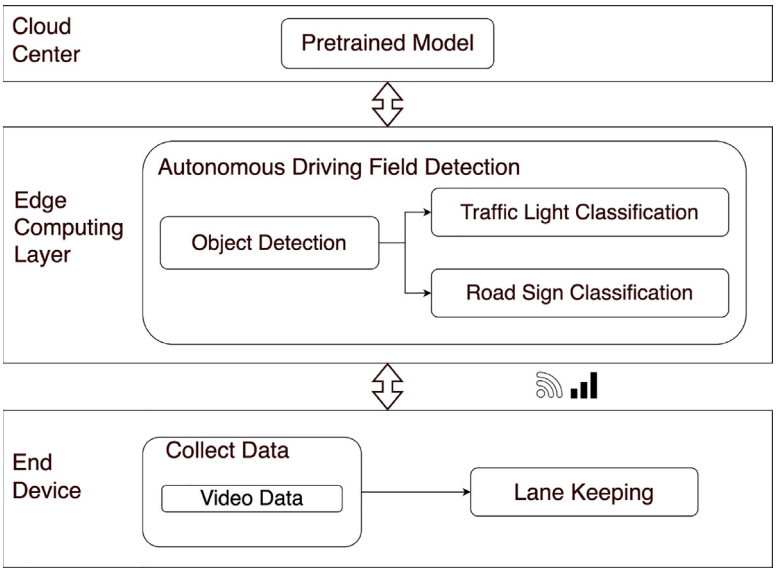

Fig. 3. The DAG model of the scenario of the autonomous vehicle in IoT–Edge–Cloud systems after simplifying.

the vehicle analysis and environment perception modules at the edge are also pruned.

3. Prune the modules whose running time is less than 1% of the total running time.
   After the analysis of the real system, the text data transmission latency and the specific processing time of the data collected by sensors, radar, and other devices occupy a very short period of time. The final task decision and control modules, which do not involve AI models, can also be completed in a very short time. Therefore, we will trim these modules.
4. Combine similar tasks that are executed concurrently if possible.
   In the traffic signal classification and road sign classification tasks, both have object detection for object localization. Thus, we merge the object detection process in these two modules. The results are sent to the subsequent tasks—traffic signal classification and road sign classification.
5. Remove the route planning module.
   Route planning is one of the most critical tasks in autonomous driving. But in creating this scenario benchmark, we remove it because the route planning task does not require real-time image data, and the panning result data transmission time is very short. This task is usually performed on a cloud server in existing real-world environments. The algorithms have been developed very maturely for the route planning task itself, and many advanced online navigation maps are available to users. Baidu's proposed Apollo autonomous driving level navigation [25] is now in use, reducing the speed of passing vehicles at intersections by 36.8%. As a result, this module can be trimmed.
6. Remove precedent and subsequent tasks of the pruning module.
   After simplifying the overall scenario according to the first five distilling rules, we will re-examine the DAG model and remove any prior or following tasks to the pruning module.

Based on the proposed six distilling rules, we prune Fig. 2 into a simplified DAG model 3. First, we merge similar modules with the same purpose. Next, we prune the parallel tasks executed on the IoT–Edge–Cloud systems at the same time. Then, we prune the modules that consume a short time, such as preprocessing and decision-making. Next, we combine similar tasks executed concurrently, such as the object classification module. At last, we removed the navigation module and related tasks.

With this simplified scenario of autonomous driving, we have scaled down the amount of code and dataset, reducing the complexity of the scenario while retaining the characteristics. Therefore, users can still evaluate systems and components in which they are interested.

4. The reference implementation of the autonomic vehicle scenario benchmark

4.1. Reference implementation

We investigated advanced algorithms and real-world datasets from academia and industry for the simplified autonomous driving scenario model proposed in the previous section. Then we implement the scenario-level benchmark for autonomous vehicles according to Fig. 3. This section briefly describes the deep learning algorithm models and datasets used in our reference implementation.

The **lane keeping** task used a CNN model [26] based on a self-attention distillation mechanism and selected CuLane [27] as a real-world dataset, which contains 3268 well-labeled training data and 358 validation data.

The **object detection** task uses YOLOv5 [28] as the deep learning network model and selected BDD100K [29] as the dataset, which contains 100,000 labeled HD datasets.

The **traffic Light classification** task uses a CNN model [30] as the deep learning network model. It uses the Nexar dataset [31] as the real-world dataset, which contains 18,659 labeled training datasets containing traffic signal images and 500,000 test data images.

The **road sign classification** task uses a CNN deep learning model [32] based on the LeNet framework [33] and the German Traffic Sign Recognition Benchmark (GTRB) [34] as the dataset, and the task classifies 43 classes of traffic signs.

4.2. A scalable IoT–Edge–Cloud benchmarking framework

In order to meet the real-world edge computing scenario, the benchmark architecture needs to consider resource allocation to deal with different sizes of AI workloads. For our simplified autonomic vehicle scenario, we propose a scalable architecture that can evaluate different sizes of systems and allocate resources (see Fig. 4).

This scalable architecture is based on Google Kubernetes [35], which allocates the offloading workloads to the edge server. On the edge server, we use TensorFlow Serving [36] to load the pre-trained models sent down from the cloud datacenter, waiting for the response to end-user tasks.

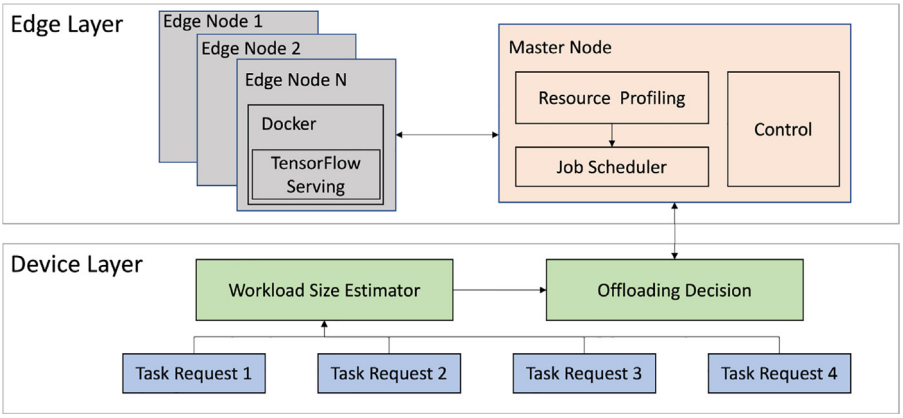

Fig. 4. A scalable IoT–Edge–Cloud benchmarking framework.

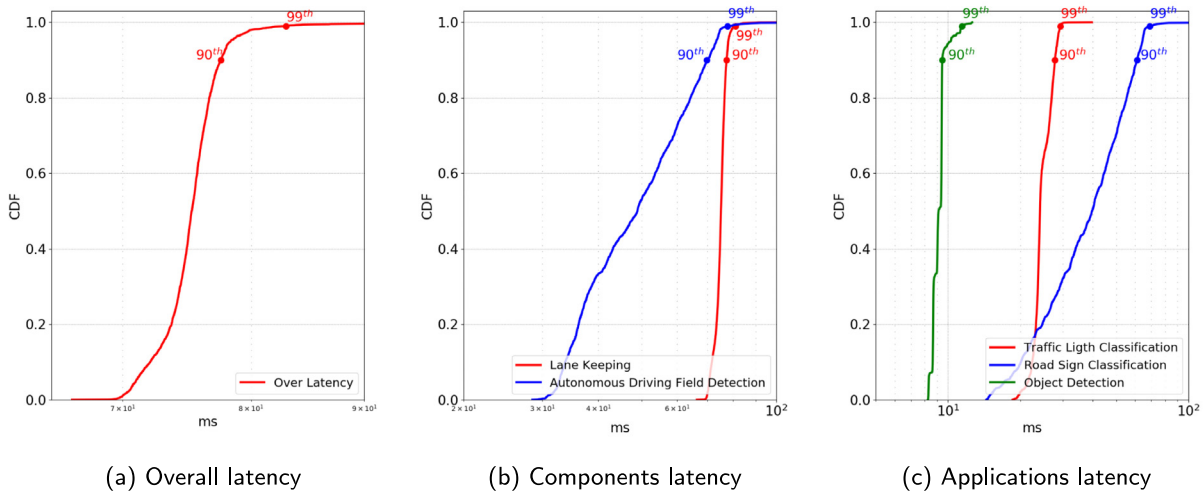

Fig. 5. Overall scenario and components latency breakdown of multi-tasks.

Table 1
Configurable parameters of the scalable framework.

Parameters	Description
The number of nodes	the number of edge computing layer nodes
AI module location	where the AI task placed: edge computing layer or end device
The number of tasks	the number of tasks that the device will send
Task size	the data input size of the task (MB)

In order to achieve system scalability, a master node is present at the edge layer to manage the computing resources and allocate workloads supplied by end devices. This architecture can scale numerous edge nodes to distribute tasks from end devices to various edge servers and computing resources. The parameters users can set are listed in Table 1.

5. Experiments and measurements

We conduct a scenario benchmark evaluation experiment based on a four-node server cluster, including one cloud server, two edge servers, and one client device. One experiment device is a CPU cloud server with two Xeon E5645 processors and 32 GB of RAM, and the other three nodes are each equipped with an Nvidia Titan XP GPU. Each node is connected to the other with a 1 GB Ethernet connection. We perform the offline training for the four AI tasks on the cloud servers and send the pre-trained model to the edge servers and end devices.

5.1. Tail latency of the whole scenario

Autonomous driving scenarios are very demanding in terms of latency, so we choose latency as a quality of service metric for this scenario benchmark. We break down the whole scenario latency to each task module to discover which modules are the primary contributors to latency in the whole scenario.

We tested 2000 autonomous vehicle task requests sent by the client device. Fig. 5 shows the latency of the whole scenario compared to the breakdown latency of each module. Since several vehicles send requests simultaneously in the real-world scenario, the tail latency metric is an important metric we need to concern about. We also pay attention to the latency data for the 90% and 99% percent of vehicle-side user queries.

Fig. 5(a) shows the end-to-end latency data for the entire scenario, with a tail latency of 76.45 ms for the 90th and 77.49 ms for the 99th percentile latency. In Fig. 5(b)(c), we have decomposed the tasks according to whether it belongs to the edge layer or the vehicle end. We can see that the overall latency of the lane-keeping task is slower than detection and classification tasks. And further decomposition of the object classification task shows that the slowest module is the road sign classification, with a tail latency of 58.92 ms for the 90th percentile latency and 67.70 ms for the 99th percentile latency. The fastest task

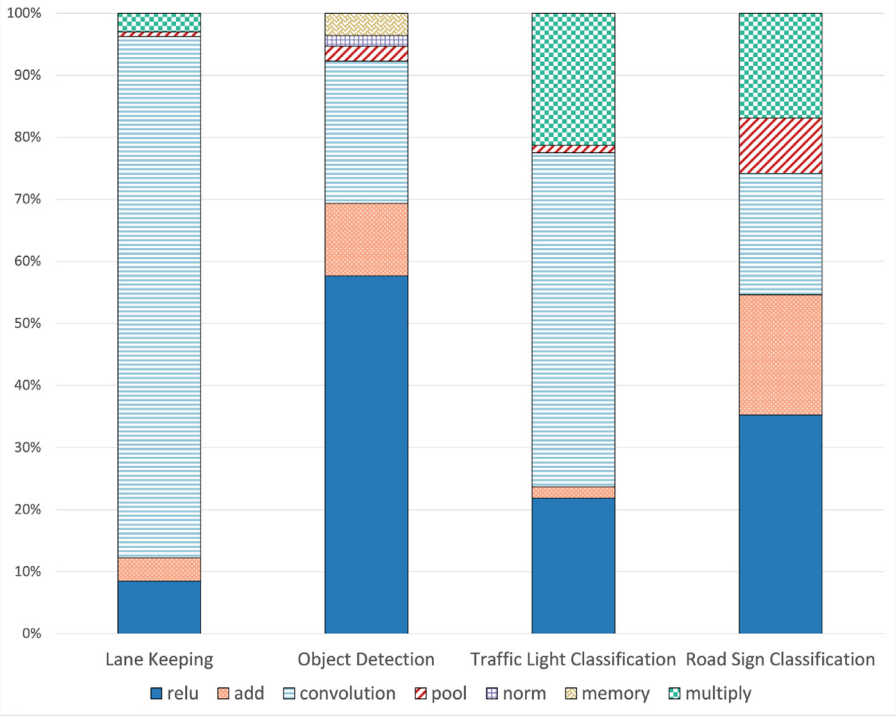

Fig. 6. Hotspots functions runtime breakdown.

is the object detection task, with a tail latency of 8.67 ms for the 90% task and 9.40 ms for the 99% task. Moreover, for the traffic signal classification, the 90th percentile latency is 23.55 ms, and the 99% percentile latency is 28.13 ms.

In our scenario benchmarking framework, the lane-keeping task is placed on the vehicle side. However, the large AI model of this task causes a slow processing speed. Therefore, it reduces the system's overall execution speed. According to the network environment's performance, users can try to place lane-keeping tasks on the edge side. An appropriate task position strategy will achieve better performance.

5.2. Hotspot function analysis

Because the majority of the tasks in autonomous driving scenarios employ deep learning models, which require the high performance of the onboard chips, as a result, we decomposed the execution time of GPUs using the profiling tool nvprof [37] offered by Nvidia. Then we analyze those hotspot functions.

We analyze each module's runtime using the nvprof tool to identify the hot functions that consume the most runtime. Then we divide these functions into seven categories based on their intrinsic computational logic: ReLU activation functions, add operations, convolution operations, pool operations, normalization, memory operations, and matrix multiplication operations.

As can be seen in Fig. 6, the convolution operations account for the most time in the lane-keeping task, which is why it is the slowest execution module. Additionally, the convolution operations take up a large percentage of all other tasks.

In object detection and road sign classification, the function with the most execution time is the ReLu activation function.

Analyzing the hotspot function is beneficial to further optimizing the CUDA library of the smart chip in autonomous vehicles. Also, for the special scenario of autonomous driving, software and hardware co-design is needed to optimize the execution speed of different modules and thus optimize the overall scenario performance.

6. Conclusion

This paper proposes the first IoT–Edge–Cloud benchmark: a scenario benchmark for autonomous vehicles. First, we analyze the challenges of creating an autonomous driving scenario benchmark. Then We reproduce the whole autonomous driving scenario picture under the IoT–Edge–Cloud system based on these user-concerned challenges and industrial-grade autonomous driving scenarios. Because of the specificity of the autonomous driving scenario, many complex factors must be considered. Therefore, the amount of code to reproduce the entire system based entirely on this scenario graph is enormous.

To resolve this issue, we propose to take a scenario benchmark view. We propose six distilling rules for simplifying the scenario of the autonomous vehicle. These rules ensure that the system's characteristics are retained while streamlining the whole system as much as possible and covering critical end-to-end IoT–Edge–Cloud paths. We obtained a simplified DAG diagram of the essential tasks from the autonomous vehicle scenario according to the distilling rules and implemented them with state-of-the-art techniques. To meet the system-level evaluation at different scales, we also implement a scalable Iot–Edge–Cloud benchmarking framework for the autonomic vehicle scenario.

Finally, we conduct several experimental evaluations of this scenario benchmark and measure the tail latency of each module. The experimental results show that lane-keeping is the most time-consuming task in the whole system. In addition, we make further analysis of the hotspot function. The result indicates that the convolution operation is the most time-consuming function. The experiment results reveal the optimization points for the software stack of autonomous vehicles.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was supported by the Strategic Priority Research Program of the Chinese Academy of Sciences, Grant No. XDA0320000 and XDA0320300.

References

[1] W. Gao, F. Tang, J. Zhan, X. Wen, L. Wang, Z. Cao, C. Lan, C. Luo, X. Liu, Z. Jiang, Aibench scenario: Scenario-distilling AI benchmarking, in: 2021 30th International Conference on Parallel Architectures and Compilation Techniques, PACT, IEEE, 2021, pp. 142–158.

[2] J. Somers, The coming software apocalypse, Atl. 26 (2017) 1.

[3] J. Zhan, A BenchCouncil view on benchmarking emerging and future computing, BenchCouncil Trans. Benchmarks, Stand. Eval. (2022) 100064.

[4] J. Zhan, Call for establishing benchmark science and engineering, BenchCouncil Trans. Benchmarks, Stand. Eval. 1 (1) (2021) 100012, http://dx.doi.org/10.1016/j.tbench.2021.100012, URL; https://www.sciencedirect.com/science/article/pii/S2772485921000120.

[5] J. Zhan, Three laws of technology rise or fall, BenchCouncil Trans. Benchmarks, Stand. Eval. 2 (1) (2022) 100034, http://dx.doi.org/10.1016/j.tbench.2022.100034, URL; https://www.sciencedirect.com/science/article/pii/S2772485922000217.

[6] T. Hao, Y. Huang, X. Wen, W. Gao, F. Zhang, C. Zheng, L. Wang, H. Ye, K. Hwang, Z. Ren, et al., Edge AIBench: towards comprehensive end-to-end edge computing benchmarking, in: Benchmarking, Measuring, and Optimizing: First BenchCouncil International Symposium, Bench 2018, Seattle, WA, USA, December 10-13, 2018, Revised Selected Papers 1, Springer, 2019, pp. 23–30.

[7] T. Hao, K. Hwang, J. Zhan, Y. Li, Y. Cao, Scenario-based AI benchmark evaluation of distributed cloud/edge computing systems, IEEE Trans. Comput. (2022).

[8] H. Blum, P.-E. Sarlin, J. Nieto, R. Siegwart, C. Cadena, Fishyscapes: A benchmark for safe semantic segmentation in autonomous driving, in: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, 2019.

[9] X. Song, P. Wang, D. Zhou, R. Zhu, C. Guan, Y. Dai, H. Su, H. Li, R. Yang, Apollocar3d: A large 3d car instance understanding benchmark for autonomous driving, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 5452–5462.

[10] J. Xue, J. Fang, T. Li, B. Zhang, P. Zhang, Z. Ye, J. Dou, BLVD: Building a large-scale 5d semantics benchmark for autonomous driving, in: 2019 International Conference on Robotics and Automation, ICRA, IEEE, 2019, pp. 6685–6691.

[11] Y. Wang, S. Liu, X. Wu, W. Shi, CAVBench: A benchmark suite for connected and autonomous vehicles, in: 2018 IEEE/ACM Symposium on Edge Computing, SEC, IEEE, 2018, pp. 30–42.

[12] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? the kitti vision benchmark suite, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2012, pp. 3354–3361.

[13] B. Maity, S. Yi, D. Seo, L. Cheng, S.-S. Lim, J.-C. Kim, B. Donyanavard, N. Dutt, Chauffeur: Benchmark suite for design and end-to-end analysis of self-driving vehicles on embedded systems, ACM Trans. Embed. Comput. Syst. (TECS) 20 (5s) (2021) 1–22.

[14] Y. Ma, Z. Wang, H. Yang, L. Yang, Artificial intelligence applications in the development of autonomous vehicles: a survey, IEEE/CAA J. Autom. Sin. 7 (2) (2020) 315–329.

[15] SAE, SAE J3016-taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles, 2021.

[16] H. Martin, K. Tschabuschnig, O. Bridal, D. Watzenig, Functional safety of automated driving systems: Does ISO 26262 meet the challenges? in: Automated Driving, Springer, 2017, pp. 387–416.

[17] N.A. Rawashdeh, J.P. Bos, N.J. Abu-Alrub, Drivable path detection using CNN sensor fusion for autonomous driving in the snow, in: Autonomous Systems: Sensors, Processing, and Security for Vehicles and Infrastructure 2021, Vol. 11748, SPIE, 2021, pp. 36–45.

[18] H. Daembkes, Automated driving safer and more efficient future driving foreword, in: Automated Driving: Safer and more Efficient Future Driving, Universität Ulm, 2017, pp. V–VI.

[19] L. Liu, S. Lu, R. Zhong, B. Wu, Y. Yao, Q. Zhang, W. Shi, Computing systems for autonomous driving: State of the art and challenges, IEEE Internet Things J. 8 (8) (2020) 6469–6486.

[20] H. Rebecq, T. Horstschäfer, G. Gallego, D. Scaramuzza, EVO: A geometric approach to event-based 6-DOF parallel tracking and mapping in real time, IEEE Robot. Autom. Lett. 2 (2) (2016) 593–600.

[21] Y. Cai, T. Luan, H. Gao, H. Wang, L. Chen, Y. Li, M.A. Sotelo, Z. Li, YOLOv4-5D: An effective and efficient object detector for autonomous driving, IEEE Trans. Instrum. Meas. 70 (2021) 1–13.

[22] J.H. Gawron, G.A. Keoleian, R.D. De Kleine, T.J. Wallington, H.C. Kim, Life cycle assessment of connected and automated vehicles: sensing and computing subsystem and vehicle level effects, Environ. Sci. Technol. 52 (5) (2018) 3249–3256.

[23] M. Simon, K. Amende, A. Kraus, J. Honer, T. Samann, H. Kaulbersch, S. Milz, H. Michael Gross, Complexer-yolo: Real-time 3d object detection and tracking on semantic point clouds, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2019.

[24] P. Sagal, Bosch seeks edge with combined software, electronics unit. URL; https://europe.autonews.com/suppliers/bosch-seeks-edge-combined-software-electronics-unit/.

[25] Baidu, Apollo, 2020, URL; https://developer.apollo.auto/.

[26] Y. Hou, Z. Ma, C. Liu, C.C. Loy, Learning lightweight lane detection cnns by self attention distillation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1013–1021.

[27] X. Pan, J. Shi, P. Luo, X. Wang, X. Tang, Spatial as deep: Spatial cnn for traffic scene understanding, 2017, arXiv preprint arXiv:1712.06080.

[28] H. Wang, Y. Xu, Y. He, Y. Cai, L. Chen, Y. Li, M.A. Sotelo, Z. Li, YOLOv5-Fog: A multiobjective visual detection algorithm for fog driving scenes based on improved YOLOv5, IEEE Trans. Instrum. Meas. 71 (2022) 1–12.

[29] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, T. Darrell, Bdd100k: A Diverse Driving Video Database with Scalable Annotation Tooling, Vol. 2, No. 5, 2018, p. 6, arXiv preprint arXiv:1805.04687.

[30] Z. Ouyang, J. Niu, Y. Liu, M. Guizani, Deep CNN-based real-time traffic light detector for self-driving vehicles, IEEE Trans. Mob. Comput. 19 (2) (2019) 300–313.

[31] V. Madhavan, T. Darrell, The Bdd-Nexar Collective: a Large-Scale, Crowsourced, Dataset of Driving Scenes, Ph. D. Thesis, Master's Thesis, EECS Department, University of California, 2017.

[32] C. Zhang, X. Yue, R. Wang, N. Li, Y. Ding, Study on traffic sign recognition by optimized Lenet-5 algorithm, Int. J. Pattern Recognit. Artif. Intell. 34 (01) (2020) 2055003.

[33] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.

[34] J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel, The German traffic sign recognition benchmark: a multi-class classification competition, in: The 2011 International Joint Conference on Neural Networks, IEEE, 2011, pp. 1453–1460.

[35] B. Burns, J. Beda, K. Hightower, L. Evenson, Kubernetes: Up and Running, O'Reilly Media, Inc, 2022.

[36] C. Olston, N. Fiedel, K. Gorovoy, J. Harmsen, L. Lao, F. Li, V. Rajashekhar, S. Ramesh, J. Soyke, Tensorflow-serving: Flexible, high-performance ML serving, 2017, arXiv preprint arXiv:1712.06139.

[37] Nvidia, Nvidia profiling toolkit, 2022, URL; https://docs.nvidia.com/cuda/profiler-usersguide/index.html.