Computers and Electrical Engineering 101 (2022) 107989

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

A synergistic reinforcement learning-based framework design in driving automation

Yuqiong Qi [a,b,*], Yang Hu [c], Haibin Wu [a], Shen Li [d], Xiaochun Ye [a], Dongrui Fan [a]

[a] SKLCA, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[b] University of Chinese Academy of Sciences, Beijing, China
[c] Electrical Engineering Department, University of Texas at Dallas, Dallas, Texas, USA
[d] National University of Singapore, Singapore

ARTICLE INFO

Keywords:
Autonomous Driving
Heterogeneous Multicore AI Accelerator
Criteria
Reinforcement Learning
Scheduling

ABSTRACT

Autonomous driving, which integrates artificial intelligence and the Internet of Things, has piqued the interest of both academics and industry because of its economic and societal benefits. Rigorous accuracy and latency requirements are important for autonomous driving safety. In order to achieve high computation performance in driving automation system, we propose in this paper a heterogeneous multicore AI accelerator (HMAI). At the same time, on the HMAI, how to allocate a large number of real-time tasks to different accelerators remains a notable problem that is worth considering. Theoretically, this problem is NP-complete, and always solved using heuristic-based and guided random-search-based algorithms. However, the global state of HMAI cannot be considered comprehensively in these algorithms, which usually leads to suboptimal allocations. In this paper, we propose FlexAI, a predictive and global scheduling mechanism on HMAI. Specifically, the proposed scheduling algorithm that is based upon deep reinforcement learning (RL). In order to evaluate the quality of strategies produced by RL agent and update the observation of the scheduling agent, two scheduling metrics are proposed: Global State Value (Gvalue), Matching Score (MS) which pays attention to the requirements of various tasks in driving automation system like emergency level. In the experimental, FlexAI achieves up to 80% execution time reduction and 99% resource utilization improvement compared with Min-min, ATA in heuristics, and genetic algorithms, simulated annealing in guided random-search-based algorithms, and unscheduled case.

1. Introduction

In the era of artificial intelligence (AI) and the Internet of Things (IoT), the autonomous vehicle industry is focused on the vertical application of AI and IoT. The realization of fully autonomous driving will undoubtedly bring benefits to both the economy and society. The autonomous driving industry is dominated by AI, combined with the traditional automotive industry, the rapid development of the fifth generation (5G) communications industry, and the mature technology of the electronics industry. Since rigorous accuracy and latency requirements are important for autonomous driving safety, current autonomous driving system should provide extreme computational resources. For example, the recently-debuted Full Self-Driving computer (FSD) from Tesla [1], is expected to be capable

* Corresponding author.
E-mail address: yuqiongqi76@gmail.com (Y. Qi).

https://doi.org/10.1016/j.compeleceng.2022.107989
Received 10 December 2021; Received in revised form 31 March 2022; Accepted 6 April 2022
Available online 19 April 2022

of processing 2300 frames per second collected from 8 surround cameras and 12 ultrasonic sensors, which is a 21 times improvement over the last generation Hardware 2.5 [2,3] in terms of image processing.

Such high demands of processing throughput need a large amount of neural network (NN)-based tasks to be processed timely in the autonomous driving system. However, traditional general processors can't efficiently process large NNs developed for Autopilot due to the lack of computational resource. Therefore, the emerging autonomous driving computing platform is designed as ASIC (e.g. Tesla FSD [1] and Horizon Journey [4]), which can focus on features of neural-network-based tasks. However, building such an accelerator-based autonomous driving computing platform is challenging.

On the one hand, with the increasing cameras and sensors (e.g. 2.5 billion pixels per second) generate overwhelming NN processing tasks, most platforms integrated one accelerator can't meet this overwhelming processing requirements. However, a future fully autonomous driving vehicle (L5) expects to integrate many more cameras/sensors and demands a performance of nearly 1000 TOPS [5]. Therefore, single accelerators need to be expanded to multiple accelerators, which can pose several new challenges for architecture design.

On the other hand, the processing workloads exposed to modern intelligent autonomous driving SoCs present increasing heterogeneity and further challenges the performance and energy of multiple accelerators. A future self-driving car can operate a variety of neural network-based tasks. Furthermore, depending on their functions, different cameras (e.g. front-facing or side-facing) can have differentiated stream generation rates and accuracy requirements.

Such composite workload flows often involve running multiple NN models with distinct layer operations and sizes. This will call for a multiple heterogeneous accelerator architecture that requires a new form of parallelism. As advocated, the performance or efficiency of future computer systems will have to rely on new accelerator-level parallelism (ALP). The ALP is defined as the parallelism of workload components (e.g. NNs for different cameras) concurrently executing on multiple accelerators. A high ALP implies that each accelerator can execute a targeted computation class faster and usually with less energy.

These observations prompt us to think about an important question: how can a driving automation computing system well-manage these challenging design aspects under rigorous performance and energy restrictions? In other words, to efficiently process such a large amount of CNN-based tasks with high variability on the complicated heterogeneous hardware substrate, effective criteria for system design that are tailored to driving automation should be defined, and efficient task scheduling mechanism should be explored to meet the criteria. Unfortunately, current computing systems for driving automation have not provided the answers for this question.

In this work, we aim to extensively explore the above system design challenges and build a comprehensive driving automation framework that synergistically handles the key design aspects. *First*, our framework features a novel heterogeneous multi-core AI accelerator (HMAI) to provide the hardware substrate for the driving automation tasks with variability. We also propose the design principle to choose the accelerators for the HMAI.

Second, our framework defines system design criteria to better utilize hardware resources and achieve increased throughput while satisfying the performance and energy restrictions. Specifically, we propose two metrics, Matching Score (MS) and Global State Value (Gvalue) to formalize the criteria.MS pays attention to the safety requirements of various tasks in driving automation systems, while Gvalue puts more weight onto the overall performance of HMAI that reflects the globality.

Finally, our framework employs a deep reinforcement learning (RL)-based task scheduling mechanism FlexAI, to resolve the task mapping issue. Specifically, we show that a robust policy can be yielded by applying deep Q-network (DQN) [7]. The RL agent in FlexAI is predictive and global. The predictive means each policy will schedule a corresponding task immediately without considering the later-coming tasks. The global feature in FlexAI means it can consider the whole performance in hardware, such as resource utilization.

2. Background and motivation

Both industry [9] and academia are intensively exploring the driving automation system. The Society of Automobile Engineers (SAE) [10] defines six levels of driving automation from Level 0 to Level 5, where Level 5 is fully autonomous. The full automation system is considered as the most multidisciplinary realm that has the most promising market value and requires the most sophisticated technology stack [6]. The current autonomous driving system mainly relies on AI-enabled automation vehicles. To deliver a functional and practical driving automation system with a safety and reliability guarantee, designers should carefully analyze and optimize every technical detail from hardware through software [7]. In this paper, we set out to explore the design principles of hardware accelerators in driving automation system and corresponding task scheduling mechanisms.

Table 1
The camera frame rates in different researches.

	Max velocity(km/h)	Frame rate(FPS)
KITTI [18]	90	10-100
ApolloScape [19]	30	30
Princeton [20]	80	10
VisLab [21]	70.9	>25
Oxford RobotCar [22]	/	11.1-16
Comma.ai [23]	/	20

2.1. Automated Cars Need Multi-Accelerators

The study in compares the number of cameras integrated in different car makers' automated vehicles [8]. According to it, we believe future automated vehicles will be equipped with more than **30** cameras

The cameras and sensors on automated vehicles can generate a massive amount of data for real-time analysis. We show the relationship between the speed of the car and the requirements of the frame rates in Table 1, which are collected from multiple studies. We can observe that the required frame rate is around 20 FPS (frames per second). Note that a high frame rate is necessary for high-speed driving [11]. In KITTI the max frame rate for a speed of 90 km/h could be 100 FPS. In industry practice, Audi sets the camera frame rate in driver assistance systems as 25 FPS [12]. The Tesla Model 3 adopts 36 FPS [13]. With the much higher safety requirements in the fully driving automation, the camera frame rates will be greater than **40** FPS in the future.

Since current single FPGA or ASIC-based accelerator cannot provide sufficient processing capacity to satisfy autonomous driving performance requirements 1200 FPS (this is calculated based on the assumption of a car with 30 cameras [13] and a generation rate of 40 FPS for each camera), multi accelerator is needed for driving automation processing.

3. Multi-accelerators call for heterogeneity

According to [14], object detection (DET), object tracking (TRA), and localization (LOC) dominate the computing of the driving automation system. While for DET and TRA, the convolutional neural network (CNN)-based computation accounts for more than 94% of the execution time. Therefore, we will focus on CNN-based tasks in the driving automation computing system. Specifically, we will focus on three typical CNN algorithms, the YOLO [28] and SSD [15] for DET, and GOTURN [16] for TRA.

Since YOLO and SSD show various achieved Average Precisions (AP) for different object areas. YOLO is good at small and medium object detection, while SSD is good at large object detection [28], [17]. Therefore, the object detection tasks in automated vehicles demand heterogeneous CNN models to ensure accuracy.

Different CNN accelerator architectures can present various advantages and disadvantages for processing diverse CNN tasks. We characterize typical perception-related CNNs (i.e., YOLO, SSD and GOTURN) on three representative CNN accelerators (AC1, AC2, and AC3) based on our CNN accelerator taxonomy, which will be elaborated in Section 5.1. We notice that each accelerator has a specific algorithm that is good at processing, as shown in Table 2. Note that the latency is critical in driving automation system [18]. Even a small performance gap can cause a severe accident. For example, consider a vehicle needs to decide whether a brake is needed based on monitoring when there is an object 200 meters away. Though it seems that the difference between the latency of AC1 and the latency of AC2 is trivial (i.e., 0.0933ms for YOLO), the aggregated difference in processing time can reach 0.6 seconds if a vehicle moves 150 meters at a speed of 100 km/h with 30 cameras in 40FPS [19]. This can cause 16 meters more break distance and will have a great impact on safety. Therefore, using heterogeneous accelerators has the potential to benefit the processing of various CNN-based tasks for video streams. Moreover, considering the driving automation system is still evolving, a heterogeneous accelerator architecture can better accommodate the ever-changing new algorithms and applications in this area [20].

4. Design challenges of driving automation system

4.1. System design criteria

To efficiently process such a large amount of CNN-based tasks with high variability on the complicated hardware substrate [21], effective criteria for system design that are tailored to driving automation should be defined [22]. Obviously the overall performance of the computing platform should be considered at first. Specifically, the execution time, energy consumption [23], and resource utilization of the platform are expected to be optimal after all tasks have been processed.

Another indispensable criterion is the safety requirement for the task processing [24]. The computing platform needs to provide differentiated processing time for the object recognition or object tracking tasks from different cameras.

For instance, the detection task of an object in front of a vehicle with a distance of 50 meters has higher priority than the object that is 80 meters away [25]. Therefore, we need to find a metric to describe whether the hardware platform's processing time for tasks from each camera is safe. A detailed discussion of system design criteria is provided in Section 6.

4.2. Scheduling mechanisms

As we know, task mapping on hardware substrate is an NP-complete problem and is generally solved using heuristic or guided random-search-based algorithms. However, the scheduling strategies based on these algorithms fail to see the global situation of

Table 2
The latency of typical algorithms in various accelerators.

	AC1 latency (ms)	AC2 latency (ms)	AC3 latency (ms)
SSD	12.20888	**12.11143**	12.20476
YOLO	**6.92883**	7.54496	6.97507
GOTURN	1.1045&	4.2073	**1.0319**

computing platform such as current resource utilization, the longest execution time among all cores, which often results in a sub-optimal allocation. An efficient task scheduling mechanism is the crux to trade-off the metrics that are defined in the system design criteria. We will elaborate our choice in Section 7.

4.3. A synergistic framework

We propose a synergistic framework for driving automation to bridge the gap between variable driving automation workloads and complicated hardware substrates, as shown in Fig. 1. Specifically, we first propose a CNN taxonomy and design principles for hardware accelerators. Based on these knowledge, we propose a novel heterogeneous multi-core AI accelerator (HMAI) to provide the hardware substrate for the driving automation tasks with variability. Our framework also defines the system design criteria to better utilize hardware resources and achieve increased throughput while satisfying the performance and energy restrictions. Specifically, we propose two metrics, Matching Score (MS) and Global State Value (Gvalue) to formalize the criteria. MS pays attention to the safety requirements of various tasks in driving automation systems, while Gvalue puts more weight onto the overall performance of HMAI that reflects the globality. Finally, our framework employs a deep reinforcement learning (RL)-based task scheduling mechanism FlexAI to meet the system design criteria.

4.4. HMAI-A heterogeneous multicore AI platform

In this section, we propose a heterogeneous multicore AI accelerator (HMAI) tailored to the CNN-related perception tasks in driving automation system. To choose the best sub- accelerators for HMAI, we firstly investigate the existing accelerator architectures for CNNs, and then describe the architecture of HMAI using three representative sub-accelerator architectures based on our CNN accelerator taxonomy.

4.5. A taxonomy for CNN accelerators

To choose the representative sub-accelerator architectures for HMAI, we need a comprehensive understanding of existing CNN accelerators as shown in Fig. 2(a). We propose a taxonomy for emerging CNN accelerators with respect to data processing style, register allocation, and data propagation types.

4.6. Data processing style

In this work, we first categorize the CNN accelerators into three styles as shown in Fig. 2 (a), namely S(ingle)conv, S(pecial)Sconv and M(ultiple)conv according to their data processing methods. As shown in Fig. 2 (b), Sconv processes a whole 2D convolution each iteration. While SSconv only processes a part of 2D convolution each iteration. For Mconv, it processes multiple 2D convolutions each iteration. We define the data processed by the accelerator in each iteration as a basic calculation unit (a.k.a., BasicUnit). For example, in Fig. 2 (b), the size of filters in a BasicUnit of Mconv is $F \times F \times T_m \times T_c$, the size of ifmaps is $I \times I \times T_c$, and the size of psums is $O \times O \times T_m$.

4.7. Register allocation

Fig. 2 (c) illustrates a high-level block diagram of a typical CNN accelerator. It consists of an accelerator chip and an external memory chip (EXMC). Processing elements (PE) array is often used as the main functional component in the accelerator chip, which contains multiply-accumulate unit (MAC) as computation units. The on-chip buffer (OCB) is used to store ifmaps, filters, and psums. We classify the CNN accelerators into two categories with respect to register type: dispersive register (DR) and concentrated register (CR). In DR the registers are dispersed in each PE, while in CR a centralized storage is used and never stores psums. The size

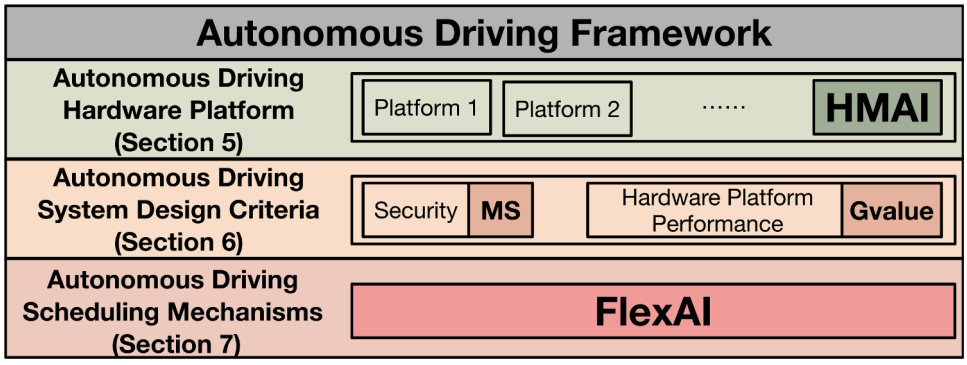


Fig. 1. An overview of Framework in Autonomous Driving System.

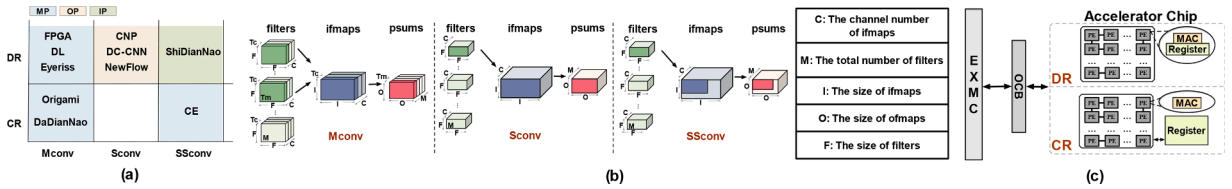


Fig. 2. (a) Quadrant Classification for CNN accelerators. (b) Data processing style. (c) Register allocation.

requirements of these registers are different among accelerators. Table 3 lists different structure designs for Sconv, SSconv and Mconv.

4.8. Data propagation types

We further define three data propagation types for the data propagation between different PEs, Ofmaps Propagation (OP), Ifmaps Propagation (IP), and Multiple Propagation (MP). (1) For OP, the ifmaps are directly sent to PEs in one BasicUnit. The filters are fixed in the PEs in advance. The psums are accumulated during the data propagation between PEs. Upon all PEs are traversed, the ofmaps neuron is generated. (2) For IP, in one BasicUnit, the filters will be sent directly to the PEs. The ofmaps are fixed in PEs. The ifmaps propagate between PEs for reuse. (3) For MP, in one BasicUnit, there will be single or multiple types of data propagation between PEs. Fig. 2 (a) shows typical CNN accelerators with different data propagation types. Note that data propagation always indicates data transfer between PEs' registers. If there is no register in each PE, data propagation means data transfer between PE array and CR, since CR is equivalent to a collection of registers in each PE.

5. The Architecture of HMAI

Based on our proposed CNN accelerator taxonomy, we propose a heterogeneous multicore AI accelerator (HMAI) that contains three representative accelerators (AC1, AC2, AC3) in an automated vehicle, as shown in Fig. 3. Each AI core has its specific architecture, as shown in Fig. 4. The CPU generates a scheduling strategy based on a well-trained RL agent for incoming perception tasks that are collected from multiple sensors. The incoming task will be directly forwarded to the target accelerator.

5.1. Why these accelerators?

The HMAI is tailored to the CNN-related driving automation perception tasks. We choose to implement all data processing styles, the Sconv, SSconv and Mconv as described in Section 5.1. To cover all data propagation types in HMAI, we further choose to implement Sconv-OP, SSconv-IP and Mconv-MP based on multiple existing accelerator types as shown in Fig. 2. To cover the register allocation methods, we implement AC1 as Sconv-OP-DR, AC2 as SSconv-IP-CR and AC3 as Mconv-MP-CR.

5.2. The architecture design of sub-accelerators

As shown in Fig. 4 (a), AC1 is based on NewFlow [34]. In AC1, each ifmaps neuron only needs to be taken from the EXMC once. In each cycle, the same ifmaps neuron is sent to all PEs, but not every PE will generate a valid signal for this ifmaps. Different filter weights are fixed in different PE's registers in advance. As to ofmaps neurons, it will be obtained after propagating to all PEs and FIFOs. The design of AC2 is shown in Fig. 4 (b), which is based on ShiDianNao [35]. In each cycle, the same filter weight is sent to all PEs. Different ifmaps neurons are read from the ifmaps register (the ifmaps register has the double buffer) to different PEs. Each PE computes only one output neuron each time. The design of AC3 is based on Origami [36], and it is shown in Fig. 4 (c), where its parameters of BasicUnit $T_m = T_c$. In ifmaps SRAM A1, the neurons of T_c channels are sent to ifmaps register. Then the register sends ($F \times F \times T_c$)-size data to the PE array. Each PE will receive ($F \times F$)-size data, while the data in A2 will be sent to the register. For filters, different $F \times F$ are sent to different PEs at each cycle until all corresponding T_m-size filters are sent. In order to guarantee the pipelining, each PE will produce a result of matrix multiplication, and then the results in all PEs will be accumulated and sent out.

6. System design criteria

In this section, we propose Matching Score (MS) and Global State Value (Gvalue), to assist autonomous driving system guide the task execution on platforms.

Table 3
Different structure design for AC1, AC2 and AC3.

	EXMC	OCB	PE Register	MAC num in each PE
Sconv & SSconv	√	x	CR or DR	1
Mconv	√	√	CR or DR	>1

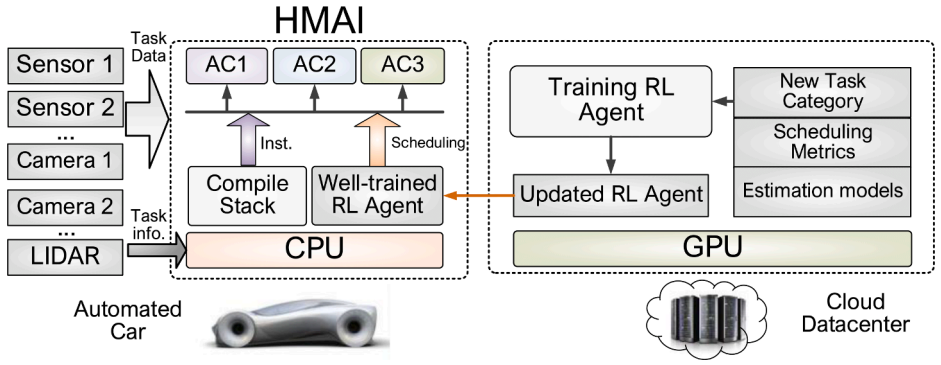


Fig. 3. An overview of HMAI.

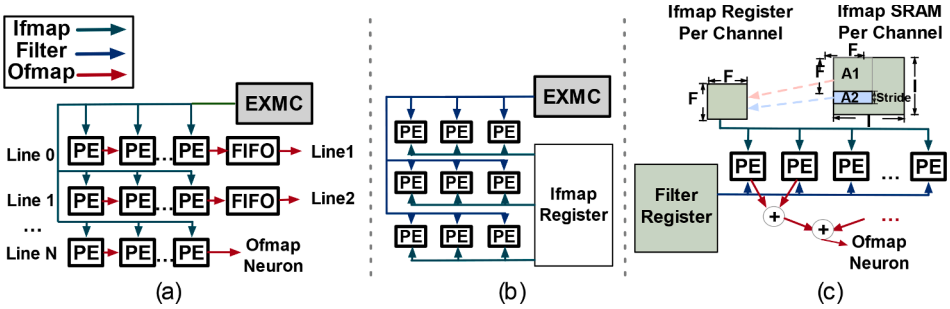


Fig. 4. (a)AC1, (b)AC2 and (c)AC3 in HMAI.

7. Matching score (MS)

Like Tesla (with 8 cameras), automated vehicles normally equip with multiple surrounding cameras to receive 360 degrees of visibility. As the different cameras have different max distance [10], each camera has its own requirement for their response time. This **response time** means automated vehicle's processing time for each camera's task. Based on the different camera's response time, we define their matching score (MS). Specifically, we characterize the camera's MS under object detection and object tracking tasks.

7.1. Matching score–Object detection

Cameras in vehicles can be divided into three categories: forward, rear and side cameras. We first introduce the MS of forward cameras. When an obstacle is detected by a forward camera, this obstacle may be in one of the three states: (1) moving in the same direction as the vehicle, (2) standing still, and (3) moving in the opposite direction as the vehicle. Among those, the obstacle in the third state needs the shortest time to be detected, and we define this shortest time as the **safety time** of this forward camera.

Based on the Responsibility-Sensitive Safety (RSS) safety model [37], the safety time of each camera can be derived. RSS reveals the relationship between safe distances and processing time of vehicles in different scenarios. When the two vehicles are driving at opposite directions, [37]proposes the minimal safe distance d_{min} between rear car c_1 and front car c_2 with velocities v_1,v_2.

$$d_{min} = \left[\frac{v_1 + v_{1,\rho}}{2}\rho + \frac{v_{1,\rho}^2}{2a_{min,break,correct}} + \frac{|v_2| + v_{2,\rho}}{2}\rho + \frac{v_{2,\rho}^2}{2a_{min,break}}\right] \tag{1}$$

In Eq. (1), ρ is the processing time of c_1, $a_{\max,accel}$ is vehicle's acceleration, $v_{1,\rho} = v_1 + \rho a_{\max,accel}$, and $v_{2,\rho} = |v_2| + \rho a_{\max,accel}$. $a_{min,break,\ correct}$ and $a_{min,break}$ are the breaking acceleration of c_1 and c_2 respectively.

In this paper, we set d_{min} to the max distance of each camera. v_1 and v_2 are the maximum velocity allowed in different areas (the maximum velocity in urban areas is 60km/h, undivided-highways is 80km/h and highways is 120km/h [38]). $a_{\max,accel}$ of c_1 and c_2 is 8.382m/s^2 which is the maximum acceleration of Tesla [10]. $a_{min,break,\ correct}$ and $a_{min,break}$ is 6.2m/s^2, which is the maximum reasonably skilled driver's braking acceleration [39]. Based on the above parameters, we can derive ρ, the safety time of forward cameras, so as to obtain the maximum response time allowed of each forward camera.

For the rear and side cameras, their safety time can be computed through Eq. (1) like forward cameras. To be noted that: (1) reversing will not be considered on the highway; (2) the maximum velocity of turning is set to 50km/h [40]. In summary, different cameras have different safety time, thus the maximum response time allowed of different cameras are different. In this paper, we define the matching score (MS) to indicate the relationship between the response time and the safety time (maximum response time) of each

camera.

In Fig. 5(a), the horizontal axis represents the object detection tasks' response time for each camera, and the vertical axis represents MS. First, we analyze the MS of the same camera in different driving scenarios.

$ST_{250FC-HW}$, $ST_{250FC-UHW}$ and $ST_{250FC-UB}$ represent the safety time of a forward camera with maximum distance 250 meters in HW, UHW and UB. We define [0-$ST_{250FC-HW}$], [0-$ST_{250FC-UHW}$] and [0-$ST_{250FC-UB}$] as accepted time (ACTime) regions, while treating [$ST_{250FC-HW}$-∞], [$ST_{250FC-UHW}$-∞] and [$ST_{250FC-UB}$-∞] as unaccepted time (UACTime) zones. If a response time for a task lies in the ACTime region, its MS grows linearly as the time increases. This is because the energy consumption of the hardware would reduce as the execution time increases while safety time is guaranteed in this region [41,79-81]. In the UACTime zone, the MS plummets to -1 due to the unacceptability of the response time. Furthermore, because the maximum velocity limit of the UB, UHW and HW is gradually increased, $ST_{250FC-UB}$, $ST_{250FC-UHW}$, and $ST_{250FC-HW}$ are gradually reduced accordingly.

Next, we introduce the MS for different cameras in the same driving scenario. As shown in Figure 5(a), $ST_{250FC-HW}$, $ST_{100RC-HW}$ and $ST_{80SC-HW}$ represent the safety time of forward camera, rear camera and side camera with a maximum distance of 250, 80 and 100 meters respectively in HW. [0-$ST_{250FC-HW}$], [0-$ST_{100RC-HW}$] and [0-$ST_{80SC-HW}$] are ACTime regions, while [$ST_{250FC-HW}$-∞], [$ST_{100RC-HW}$-∞] and [$ST_{80SC-HW}$-∞] are UACTime zones. The trend of MS for these three cameras in ACTime and UACTime are the same as above.

7.2. Matching score–Object tracking

This section will introduce cameras' MS when its task type is object tracking. In the Fig. 5(b), ST_{OD} and ST_{OT} is the safety time of the same camera when its task type is object detection (DET) and object tracking (TRA) respectively. In autonomous driving system, TRA follows DET to predict the trajectories of moving objects, which indicates that TRA is processed after DET for the same image. Therefore, ST_{OT} should not be less than ST_{OD}, and we set ST_{OT} equals to ST_{OD} here. In the Figure 5(b), [0-ST_{OT}] is ACTime, and [ST_{OT}-∞] is UACTime. When TRA's response time of the current camera is in ACTime, MS is always -1, otherwise MS is 1.

8. Global state value

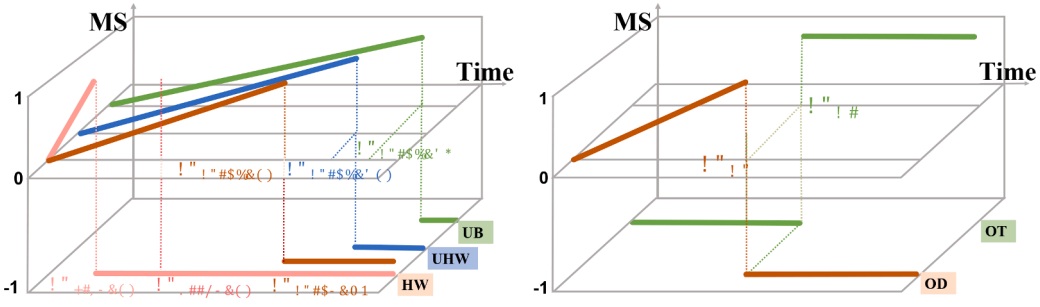
To evaluate the overall performance of HMAI, we consider energy consumption E, runtime T and resource utilization balance rate *R_Balance*. *R_Balance* means the balance of resource utilization in HMAI, thus the higher *R_Balance*, the less idle accelerators in HMAI at every moment. Whenever HMAI completes processing a task, these three values change accordingly. As the energy consumption of HMAI is expected to be as small as possible, the shorter the running time and the better the resource utilization balance rate to be, we define the Global State Value as $\text{Gvalue} = (-\text{E} - \text{T} + \textit{R\_Balance})/3$(after normalization).

9. FlexAI-A Task scheduling engine

In the autonomous driving system, the dynamic environment can generate a massive amount of tasks, while the hardware resources are limited. Thus based on the metrics in criteria, how to designate tasks to different accelerators in HMAI needs to be carefully designed.

We use a real case to show the necessity of scheduling. Consider that when 30 cameras in a vehicle work once, then 30 frames will be generated simultaneously, thus we assume there will be 30 SSD tasks to process. We cannot just allocate the same task to its best-fit accelerator because this will hurt the resource utilization of HMAI and overwhelm the chosen accelerator. Therefore, future driving automation platform needs an efficient task scheduling mechanism to trade-off among execution time, energy consumption, resource utilization and matching score.

The scheduling problem faced by HMAI is NP-complete. Conventional algorithms used to solve it can be classified into two groups, the heuristic-based and guided random-search-based algorithms. As for heuristic-based algorithms, proposed the Adaptive Task-partitioning Algorithm (ATA) to find out the scheduling policy of a task to consume as little energy as possible while guaranteeing



(a) MS for object detection (b) MS for object tracking

Fig. 5. MS in automated vehicles.

the latency. The Min-min algorithm is considered as optimal; however, this algorithm can only consider the best hardware for each task while neglecting the global performance of HMAI.

Genetic algorithms (GAs) and simulated annealing (SA) are the most popular and widely used techniques for task scheduling problems in guided random-search-based algorithms. However, a fitness equation in GA and a cost function in SA are needed to select the optimal strategy for current tasks, thus the global performance like resource utilization of HMAI can't be taken into account.

In this paper, we propose FlexAI, a learning-based task scheduler to resolve the scheduling issues in driving automation system. Table 4 compares our work with other algorithms with respect to the coverage of the metrics proposed in Section 6. Specifically, to perceive the global performance in HMAI, we will use deep reinforcement learning (RL) in FlexAI as the scheduling algorithm introduced in Section 7.1. In Section 7.2, we will introduce how to use scheduling metrics to get the reward in RL.

10. How the RL agent works

We propose a reinforcement learning (RL)-based algorithm for task scheduling on the HMAI. A RL agent can learn strategy by interacting with the environment without any supervision. In each episode of learning, the agent can provide decision-making policies according to the current environment (HMAI) and the long-term objective. This is done by receiving feedback in form of a reward from the environment.

Assuming there are N CNN accelerators $\{H_1,H_2,\ldots.H_N\}$ in the HMAI, and there will be M tasks $\{A_1,A_2,\ldots.A_M\}$ coming in sequence, of which each is a CNN-based task like object detection based on YOLO or SSD, object tracking based on GOTURN. Then, the proposed RL algorithm generates scheduling strategy $P = \{p_1,p_2\ldots p_M\}$, each of which indicates the task A_i will be scheduled to H_j under the guidance of p_i. When A_{i} is executed, the metrics of HMAI will be updated accordingly, and the difference between the updated value and the previous value is denoted by reward r_i, thus the corresponding reward set is $\{r_1,r_2\ldots r_M\}$.

The RL agent input contains three parts when training: (1) Task-Info including three parameters: **Amount**: the computation amount when task is processed; **LayerNum**: the CNN layer's number in the task; **safety time**: described in the Section 6.1. (2) HW-Info: the current information of all accelerators in the HMAI (the parameters of HW-Info will be described in Section 7.2). (3) Reward, in inference, there is no reward because the network doesn't need to update.

In this paper, we will use DQN to learn scheduling strategy from episodic job queues. [53] divided the RL algorithm into three categories: critic-only (e.g. DQN [54], actor-only (e.g. Policy gradient [55]) and actor-critic (e.g. DDPG [56]). When the scheduling strategy of a single task is generated, the critic-only algorithm can be trained once directly. However, the actor-only algorithm can only be trained after the scheduling strategies of all tasks in a task queue are generated. Since the number of tasks in each task queue in autonomous driving system is extremely large (up to 30,000 introduced in Section 8.3), to reduce training time, we will not choose actor-only category. Furthermore, although actor-critic can be trained in the same way as critic-only, due to its high computation complexity, we will choose DQN that falls into the critic-only category.

In our method, the two networks are denoted by EvalNet $D_1$with the parameter θ_1 and TargNet D_2 with the parameter θ_2. EvalNet is used to generate the scheduling policy of the current task, while TargNet is used to update the parameters of the EvalNet. These two networks are consisting of two fully connected layers, and their input S_i is Task-Info and HW-Info of task A_i. The output of these two networks is a group of Q values Q_j. Q_j is the cumulative value of the reward: $Q_j = \sum_{n=i}^{M} r_n$, which is generated after the task A_i was executed on the H_j. After obtaining N Q values, EvalNet or TargNet will choose H_j which attains the maximum Q value after A_i is executed. The choice $H_j$of EvalNet or TargNet is a scheduling strategy p_i which guides to schedule task A_i.

Figure 6 shows the working process of our RL scheduling agent. First, EvalNet D_1 will generate scheduling strategy for the input S_i, and use it to allocate task A_i to H_j. Then, in training, (1) HMAI in cloud uses H_j to update HW-Info, and calculates reward r_i. Next, HW-Info will combine with Task-Info of next task $A_{i+1}$to generate S_{i+1}; (2) the record (S_i, H_j, r_i, S_{i+1}) is saved in memory, and if the total amount of records in the memory is greater than the memory size at this time, the RL agent will use $record_m$ -$record_n$ to start learning. (3) in learning, as for $record_i$, EvalNet D_1 will use S_i to generate Q_i, and TargNet D_2 will use S_{i+1} to generate Q_{i+1}. Then $\theta_1$is updated by minimizing the loss:$L = (y_i - maxD_1(s_i|\theta_1))^2$,where $y_i = r_i + \gamma maxD_2(s_{i+1}|\theta_2)$. The parameter θ_2 in D_2will be copied directly from D_1 every fixed time. In inference, HMAI in vehicle uses H_j to update HW-Info, and then S_{i+1} is sent to the EvalNet D_1 directly.

11. The way to get reward

In this section, we will introduce how to use scheduling metrics to get the reward in RL. Suppose that the information (Info) of H_i is $(E_i, T_i, R\_Balance_i, MS_i)$, and we use Info for each accelerator in HMAI to constitute the HW-Info. After A_j is scheduled to H_i, the energy consumption, time, MS and resources utilization balance rate of processing A_j are denoted by e_j, t_j, ms_j and r_j. Thus, for H_i, (1)$E_i += e_j$;

Table 4
The metrics of some algorithms and FlexAI.

Metrics	Heuristic			Random-search	FlexAI		
	EDP [50]	Min-Min [44]	ATA [43]	W-rand [51]	GA [42]	SA [48]	
Time	√	√	x	√	√	√	√
Energy	√	√	√	√	√	√	√
Resrc	x	x	x	x	x	x	√
MS	x	√	√	x	√	√	√

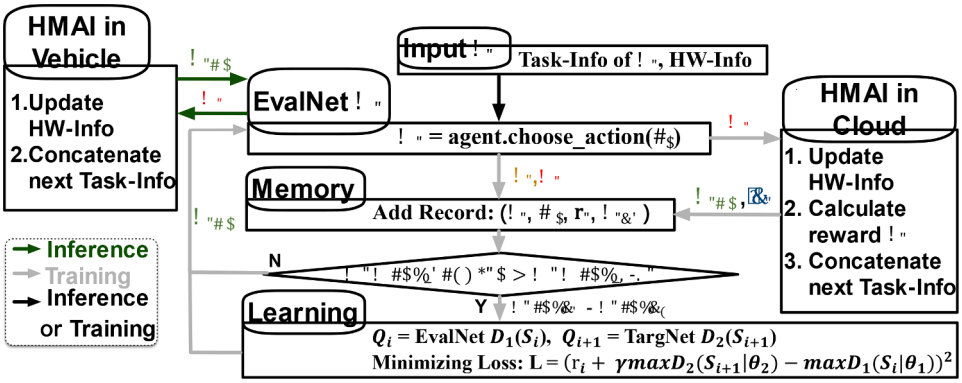


Fig. 6. The work flow of the RL scheduling agent.

(2)$T_i += t_j$; (3)$MS_i += ms_j$; (4) $R\_Balance_i = \frac{r_j + R\_Balance_i}{num}$ (num is the number of tasks has been executed in H_i,). Until now, the energy consumed in each accelerator is $\{E_1, E_2, \ldots E_N\}$ respectively, the total time is $\{T_1, T_2, \ldots T_N\}$, the resource utilization is $\{R\_Balance_1, R\_Balance_2, \ldots R\_Balance_N\}$, and the sum of the MS in each accelerator is $\{MS_1, MS_2, \ldots MS_N\}$. Then for HMAI, (1) $E = \sum_{i=1}^{N} E_i$; (2)$R\_Balance = \frac{1}{N}\sum_{i=1}^{N} R\_Balance_i$; (3)$MS = \sum_{i=1}^{N} MS_i$;(4)$T = max\{T_1, T_2, \ldots T_N\}$.

Now if there are currently $M-1$ tasks scheduled to HMAI, at this moment, the HMAI has E, T, $R\_Balance$, and MS. When the M_{th} task is executed, the four values will be updated to E_{new} T_{new}, $R\_Balance_{new}$, and MS_{new} Then, after processing M_{th} tasks reward is given by $(-E_{new} - T_{new} + R_{Balance new})/3 - (-E - T + R\_Balance)/3 + MS_{new} - \text{MS} = Gvalue_{new} - Gvalue + MS_{new} - \text{MS}$.

12. Evaluation

12.1. The dynamic driving environments

12.1.1. Parameters

To simulate a variety of vehicle driving areas and scenarios, we define several parameters, which characterize dynamic driving environments. When the parameter for driving area(urban areas (UB), undivided-highways (UHW), highways (HW)) changes, the frequency of cameras (Camera_HZ), the maximum number of turning (MaxTimes_Turn) and reversing (MaxTimes_Reverse), the longest duration of turning (MaxDuration_Turn) and reversing (MaxDuration_Reverse), the speed of the vehicle (Velocity), and the safety time of cameras (Safety_Time) will all change. Moreover, when the parameter for camera function type (FC, FLSC, RLSC, FRSC, RRSC and RC) changes, Camera_HZ and Safety_Time will vary. Similarly, when the parameter for driving action (Go straight, Turn and Reverse) changes, Camera_HZ will change also.

12.1.2. Task Queue

In this experiment, we use images in the KITTI object tracking 2D dataset as tasks in our task queue. This object tracking dataset consists of multiple sequences, and the images in each sequence are the continuous outputs for one camera in our vehicle. Based on the above parameters, we create different driving routes with various driving distances, and all tasks generated by the vehicle during the route form one task queue. In addition to the dataset that constitutes the task queue, we also specify the task amount at different time in this task queue. Fig. 7 illustrate an example of a task queue when a vehicle has a 160m route in UB and its velocity is 20m/s. Parameters such as Camera_HZ, Safety_Time are derived from Table 5. As mentioned in Section 2, we alternately use YOLO and SSD to process the

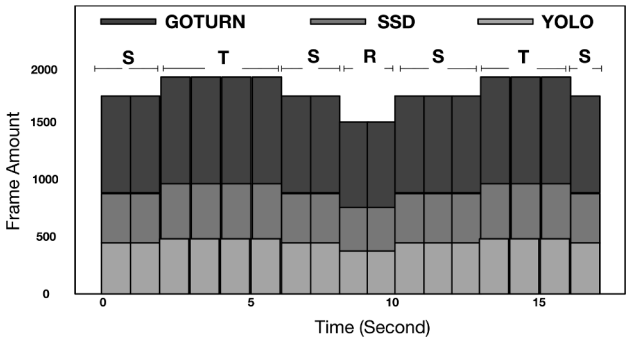


Fig. 7. The task queue.

DET tasks for each camera, and use GOTURN to process TRA tasks, thus the task types in Fig. 7 are YOLO, SSD and GOTURN. In Fig. 7, S, T and R indicate three scenarios: going straight, turning and reversing, and the start time and lasting time of each scenario is randomly determined.

13. Performance Analysis for HMAI

13.1. The construction of homogeneous and heterogeneous platforms

Based on 30 cameras and a generation rate of 40 FPS for each camera, autonomous driving system will generate 1200 frames per second. Since YOLO and SSD are alternately used to process the DET tasks for each camera, and GOTURN is used to process TRA tasks, we assume that there are 600 SSD, 600 YOLO and 1200 GOTURN should be processed per second. In Fig. 8, three homogeneous platforms and four heterogeneous platforms are given, which can satisfy above performance requirements. The construction principle of these platforms is: not only to reduce the waiting time of all tasks, but also to reduce the idle time of all accelerators in the platform as much as possible.

When processing 600 SSD, 600 YOLO and 1200 GOTURN, the resource utilization of seven platforms are given in Fig. 8. We can find that, compared with homogeneous platforms, all heterogeneous platforms can achieve the higher resource utilization. Moreover, among all heterogeneous platforms, (4 AC1, 4 AC2, 3 AC3) can always achieve the highest resource utilization. Therefore, we choose to use it to construct our heterogeneous platform HMAI.

13.2. Experimental methodology

The performance and energy of HMAI is measured by the following tools. For the performance evaluation, a customized cycle-accurate simulator was designed and implemented to measure execution time in number of cycles. This simulator models the micro architectural behavior of each hardware module of our design. In addition, we use ARM1176 as the main control processor in the HMAI to do task scheduling.

For measuring area and power, we implemented a Verilog version of each hardware module, then synthesized it. We used the Synopsys Design Compiler with the TSMC 12nm standard VT library for the synthesis, and estimated the power consumption using Synopsys PrimeTime PX. In addition, the design of AC1, AC2 and AC3 is based on [34] [35] [36]. And the SRAM is generated by Synopsys Memory Compiler and the interconnect bus is generated by Synopsys DesignWare AMBA IP.

13.3. Baseline

To compare HMAI with state-of-the-art work, we evaluate HMAI with NVIDIA Tesla T4 GPU, and above three homogeneous platforms.

13.4. Experimental methodology

Here, we create different driving routes for urban area (UB) with various distances from 1km to 2km, and vehicle's velocity is set to 60km/h. In Experimental, first, 5 task queues are constructed, and then we use HMAI including FlexAI, NVIDIA Tesla T4 and three homogeneous platforms to process each task queue.

13.5. Experimental results

Fig. 9 shows the TOPS/W of HMAI and the homogeneous platforms, normalized to Tesla T4. HMAI improves TOPS/W by 11%, 84% and 15% compared to three homogeneous platforms on average. The main reason is the reduction of a large number of redundancy computing resources. Moreover, HMAI has higher TOPS/W than Tesla T4. The reason for that is HMAI can be considered as an accelerator dedicated to processing perception tasks in autonomous driving, while GPU is a general processor.

14. Performance analysis for FlexAI

In this section, we set up experiments to compare our RL-based FlexAI with other state-of-the-art schedulers.

Table 5
The parameters.

Parameter Setting		Current Setting	
MaxTimes_Turn	10	Turn Times	2
MaxTimes_Reverse	10	Reverse Times	1
MaxDuration_Turn	10	Turn Duration	3s, 4s
MaxDuration_Reverse	20	Reverse Duration	2s

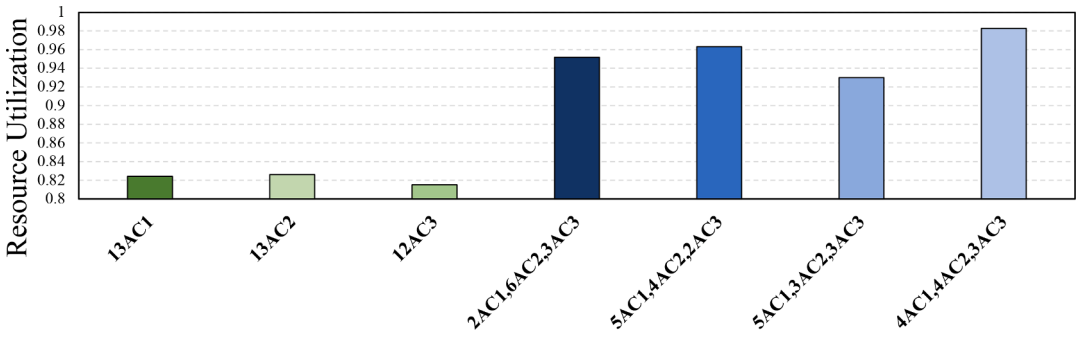


Fig. 8. Comparison between homogeneous and heterogeneous platforms.

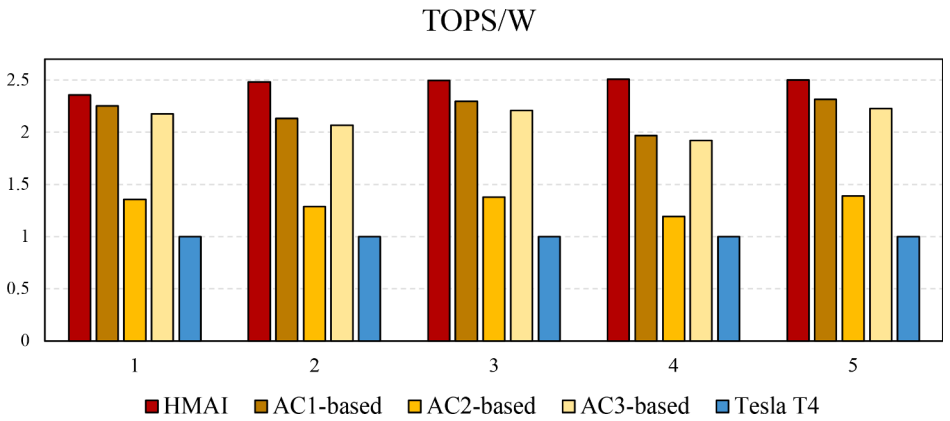


Fig. 9. Comparison between HMAI and baselines.

14.1. Baseline

We use Min-Min, ATA in heuristics, GA, SA in guided random search techniques, as well as the unscheduled worse case as our baselines.

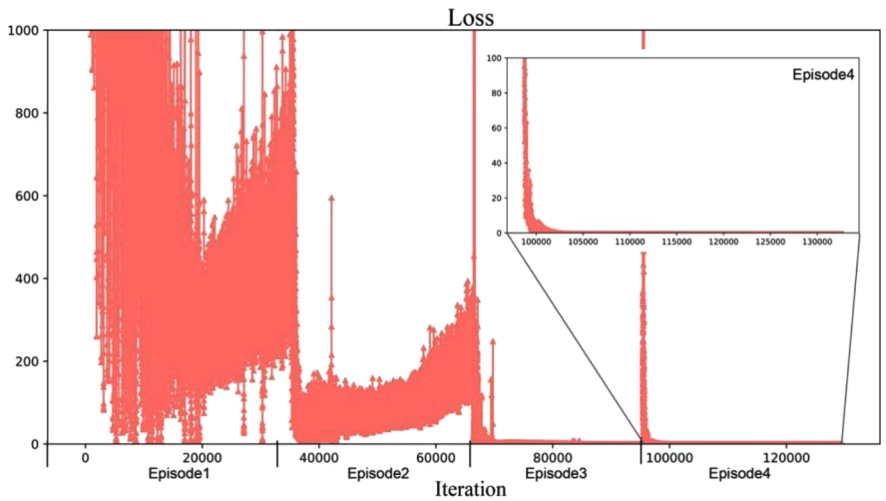


Fig. 10. Loss curve of RL agent. The threshold of the y-axis is 1000.

14.2. The parameters for constructing task queue

We create different driving routes for urban area (UB), undivided-highways (UHW), and highways (HW) with various distances from 1km to 2km, and velocity is set to 60 km/h, 80 km/h and 120 km/h, respectively.

14.3. Training

The DQN used in FlexAI agent includes two networks with exactly the same structure but different updating paces. Each network is comprised of two fully connected layers, and a softmax layer. The number of neurons of the fully connected layers are 256 and 64 with ReLU non-linearity. We train three RL agents for UB, UHW, and HW, respectively. Each agent is trained on the NVIDIA TITAN-XP with 1000 episodes, and each episode includes one task queue. The learning rate for training the EvalNet is 0.01.

14.4. Training Loss Curve

Fig.10 shows the training loss curve of FlexAI RL agent in urban area. Each iteration represents one task, and each episode contains one task queue. As all object detection and object tracking tasks generated by a vehicle in a 1km - 2km route will form one task queue, each episode will contain up to 30,000 tasks. In Fig. 10, the loss of the second episode gradually stabilizes after 10,000 iterations, while in the third and fourth episodes, except for the loss of the initial 2,000 iterations, the subsequent loss gradually tends to 0. The reason for that is the composition of tasks in each episode are very similar, thus the network trained in prior episodes will be applicable to subsequent episodes. This also further illustrates that if the task types do not change, the well-trained RL agent can be used all the time in automated vehicles.

14.5. Experimental methodology

For each area, first we use well-trained agent of FlexAI and each baseline to generate the scheduling strategy for 5 task queues. Next we compare the metrics between FlexAI and baselines for each task queue.

14.6. Experimental results

The time in Fig. 11 includes three parts: (1) scheduling strategy runtime in CPU, (2) task waiting time and (3) task execution time. In Fig. 11(a), FlexAI can maximally reduce the time by 60%, 88%, 33%, 36% and 87% compared to ATA, GA, Min-Min, SA and worse case in urban area. And for the geometric mean, FlexAI decreases at most 87%. The reason for this is FlexAI can effectively reduce the task waiting time, and more details can be found in Section 8.4. For Min-Min, SA and ATA, they perform close to the FlexAI does since they consider execution time when scheduling. However, due to the fact that GA's performance is affected by the selection of the initial population, its time is much large than FlexAI. To summary, FlexAI can always achieve the minimum time in three areas, and this will ensure the safety of autonomous driving.

As shown in Figure 11(b), the *R_Balance* of FlexAI has been maximally improved by 837%, 957%, 62%, 55% and 960% compared to ATA, GA, Min-Min, SA and worse case in urban area. As for the geometric mean in all areas, *R_Balance* in FlexAI is always the best. This is because among all scheduling strategies, only FlexAI considers the balance of resource utilization. The situation of *R_Balance* in the other two areas is the same as the urban area. In HMAI, by increasing *R_Balance*, the task waiting time can be reduced, and at the same time it can decrease the waste of the hardware resources and improve the vehicle's endurance.

In Fig. 11(c), MS in FlexAI is larger than that in GA, Min-min, SA and worse case (up to 1.02, 1.12, 0.83, 1.32), however smaller

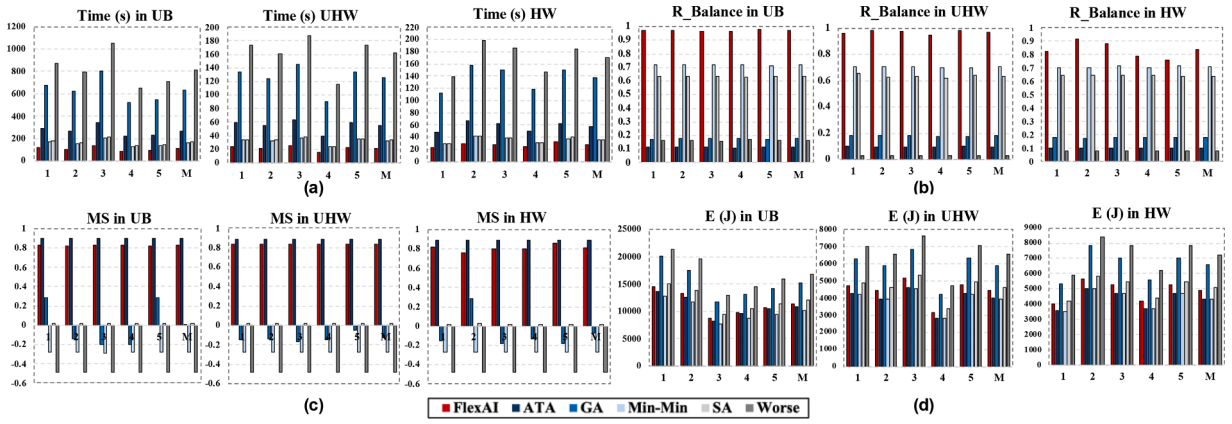


Fig. 11. Comparison between FlexAI and baselines. UB, UHW and HW means urban areas, undivided-highways and highways. M is the geometric mean of 5 task queues.

than that in ATA in the urban area. The reason is ATA is optimized towards MS, but FlexAI needs to tradeoff among four metrics. In addition, except for ATA and FlexAI, the other baselines' MS are always less than 0, which means there are many tasks in each task queue whose processing time are greater than the safety time. The situation of MS in the other two areas is the same as the urban area. In autonomous driving, higher MS represents better safety, and more discussion can be found in Section 8.4.

Fig. 11(d) shows the comparison of energy. Although FlexAI can achieve lower energy than GA, SA and worst case in all areas, it is slightly higher than the others. Some reasons are as follow. First, energy-performance tradeoff is common in accelerators. Moreover, FlexAI needs to consider T, E, *R_Balance* and MS at the same time, which makes tradeoff more difficult.

15. Autonomous driving metrics

As we mentioned in Section 6.1, since each camera in a vehicle has a corresponding safety time, each task will have its safety time according to the camera that generated it. In this section, we define safety time meet rate (STMRate) to describe the proportion of tasks in a task queue whose processing time is less than its safety time. In Fig. 12, for each task queue, the STMRate of FlexAI is basically close to 100%, which means the processing time of almost all tasks can ensure the driving safety. The reason for that is FlexAI considers MS when generating scheduling strategies, and MS indicates the relationship between the task processing time and task safety time. Here, since ATA is optimized towards MS, the STMRate of each task queue is also very high under ATA.

15.1. Braking Distance

Here, we assume that after a vehicle moves 1km, its forward camera finds there is an object 250 meters away, so it needs to take braking immediately. The current velocity of the vehicle is 60 km/h, and the braking deceleration is 6.2 m/s^2. In Fig. 13 (a)(bar), the breaking distances under different schedulers are presented.

As shown in Fig. 13 (a)(bar), except for GA, the breaking distances of other schedulers are all less than the safe distance 250 m, and FlexAI has the smallest breaking distance 47.08 m. Since the breaking distance is strongly related to the braking time (breaking distance is calculated based on the Eq. (1) in Section 6), FlexAI has the smallest breaking time, as shown in Fig.13 (a) (blue line).

The total breaking time breakdown of each scheduler is shown in Table 6. T_{wait} is the waiting time of the current task (the current task is used to detect the object that caused the braking); $T_{schedule}$ represents the runtime of each scheduler; $T_{compute}$ is the processing time of the current task in HMAI; T_{data} is the time which is used to transmit the control commands to the vehicle's actuator through the Controller Area Network (CAN) bus, which is 1 ms in this vehicle [57]; T_{mech} is the time that mechanical components of the vehicle takes to start reacting, which is 19 ms. In Table 6, T_{wait} in FlexAI is 0, while T_{wait} is much larger than the time of other parts in other schedulers. Therefore, although $T_{schedule}$ and $T_{compute}$ of FlexAI are not the best, the total breaking time of FlexAI is the smallest.

Note that in our experiment, the vehicle generates a task queue from starting to braking. As shown in Figure 13 (b), *R_Balance* in FlexAI is the largest. It means under FlexAI, the resource utilization in HMAI is the most balanced, thus the number of idle accelerators in HMAI at every moment is the least as well. For instance, if task A has the fastest execution time in the accelerator A and accelerator A is currently busying, FlexAI will schedule task A to other idle accelerators to reduce its waiting time, while for Min-Min, task A will be waiting until accelerator A changes to idle. Therefore, for FlexAI, since it has the highest *R_Balance*, its T_{wait} in Table 6 is the smallest. And under the smallest T_{wait}, FlexAI has the smallest braking distance in Figure 13 (a)(bar).

16. Related work

To the best of our knowledge, an emerging line of research has found that reinforcement learning proves to be effective in solving scheduling problems in various domains. uses RL to make real-time decision for the scheduling problem in flying mobile edge computing platform, and the reward of this RL agent includes the energy consumption of all user equipment.proposes a multi-agent reinforcement learning approach for job scheduling in grid computing, of which the reward consists of the total execution time of all jobs. Uses RL to deal with the radar resource management problem when the radar assigns limited time resource to a set of tasks, and the reward is comprised of the number of tasks delayed or dropped. For mobile-edge computing system.

As the rewards of the algorithms aforementioned are designed for their specific scenarios, these ad-hoc formulations cannot be used to solve the scheduling problem in autonomous driving. In autonomous driving systems, each CNN-based task should be handled separately, and the reward should take into account not only the dynamic changes of the environment (HMAI), such as current resource utilization, but also the total energy consumption and the longest execution time among all accelerators. Furthermore, whether the current strategy meets the real-time requirements of the cameras in autonomous vehicles also matters. Therefore, it is desirable to develop RL-based scheduling algorithms for autonomous driving.

17. Conclusion

By exploring the variability of workloads and performance requirements in driving automation and the heterogeneity of multi-accelerators, we purpose a comprehensive framework that synergistically handles the heterogeneous hardware accelerator design principles, system design criteria, and task scheduling mechanism. First, based on a taxonomy for emerging CNN accelerators, we design a heterogeneous multicore AI platform (HMAI) which adopts three typical CNN accelerator architectures. Next, by designing two metrics: Matching Score and Global State Value, autonomous driving system can guide the task execution on the platform. Finally, FlexAI-a reinforcement learning-based mechanism are proposed to generate scheduling policies in autonomous driving.

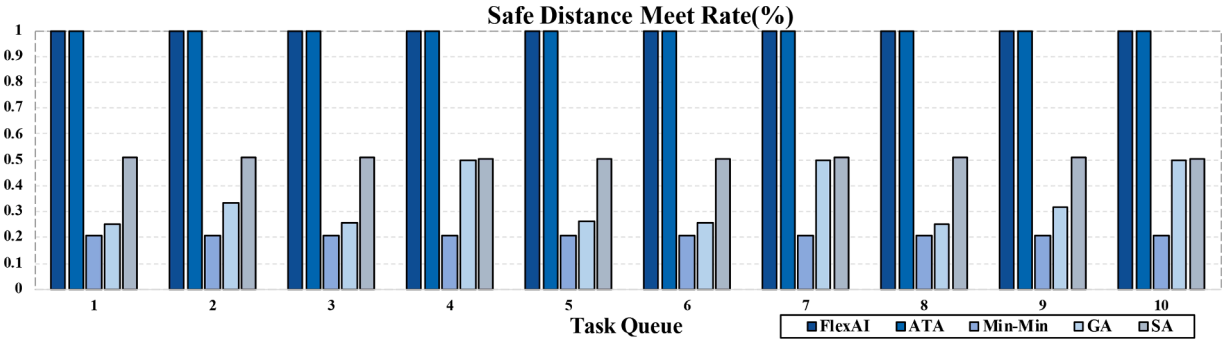


Fig. 12. Safe distance meet rate (STMRate).

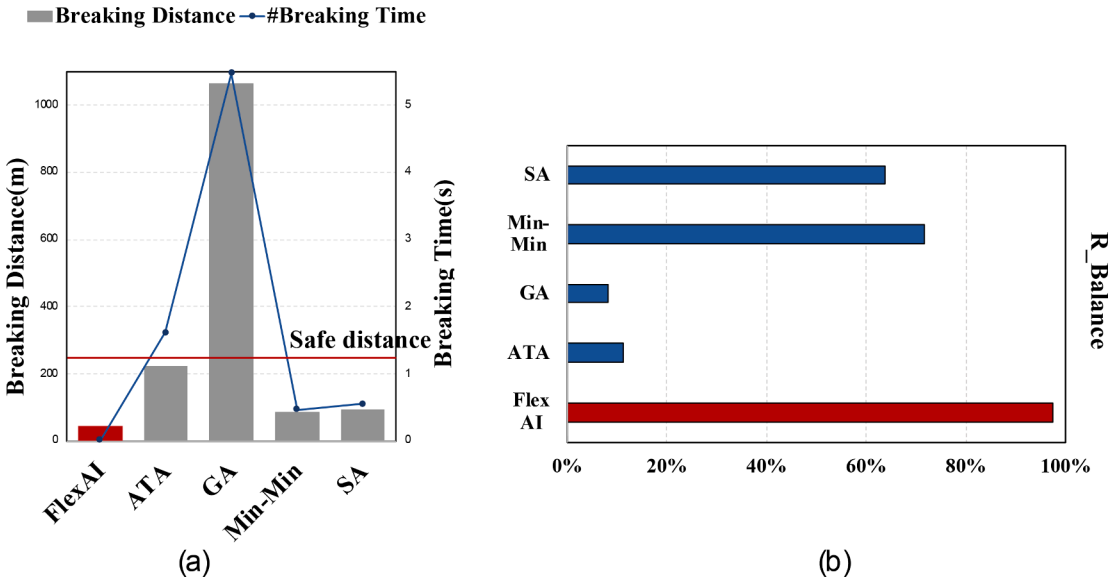


Fig. 13. (a)Breaking distance and total breaking time; (b) *R_Balance*.

Table 6
Total breaking time breakdown.

	T_{wait} (s)	$T_{schedule}$ (s)	$T_{compute}$ (s)	T_{data} (s)	T_{mech} (s)
FlexAI	0	0.001	0.00754	0.001	0.019
ATA	1.60452	5.32E-05	0.00587	0.001	0.019
GA	5.46729	1.05E-05	0.00754	0.001	0.019
Min-Min	0.4551	1.32E-05	0.00587	0.001	0.019
SA	0.54428	2.68E-05	0.0067	0.001	0.019

Declaration of Competing Interest

None.

Funding

This work was supported by the National Natural Science Foundation of China (Grant No. 61732018, 61872335, and 61802367), the Strategic Priority Research Program of Chinese Academy of Sciences (Grant No. XDC05000000), and the International Partnership Program of Chinese Academy of Sciences (171111KYSB20200002).

References

[1] Broggi A, Buzzoni M, Debattisti S, Grisleri P, Laghi MC, Medici P, Versari P. Extensive tests of autonomous driving technologies. IEEE Trans Intell Transp Syst 2013;14(3):1403–15.
[2] Driving A. Levels of driving automation are defined in new sae international standard j3016: 2014. Warrendale, PA, USA: SAE International; 2014.
[3] Geiger A, Lenz P, Urtasun R. Are we ready for autonomous driving? The kitti vision benchmark suite. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE; 2012. p. 3354–61.
[4] Yu B, Hu W, Xu L, Tang J, Liu S, Zhu Y. Building the computing system for autonomous micromobility vehicles: design constraints and architectural optimizations. In: 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE; 2020.
[5] Chen C, Seff A, Kornhauser A, Xiao J. Deepdriving: learning affordance for direct perception in autonomous driving. In: Proceedings of the IEEE International Conference on Computer Vision; 2015. p. 2722–30.
[6] Augonnet C, Thibault S, Namyst R, Wacrenier PA. Starpu:a unified platform for task scheduling on heterogeneous multicore architectures. Concurr Comput: Practic Exp 2011;23(2):187–98.
[7] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello, and.
[8] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A.C. Berg,Dssd: Deconvolutional single shot detector, arXiv preprint arXiv:1701.06659, 2017.
[9] Held D, Thrun S, Savarese S. Learning to track at 100 FPS with deep regression networks. In: Computer Vision–ECCV 2016 - 14th European Conference. Proceedings; 2016. October 11-14Part I, 2016, pp. 749–765.
[10] Bertsimas D, Tsitsiklisetal J. Simulated annealing. Stat Sci 1993;8(1):10–5.
[11] Oh E, Han W, Yang E, Jeong J, Lemi L, Youn C. Energy-efficient task partitioning for CNN-based object detection in heterogeneous computing environment. In: International Conference on Information and Communication Technology Convergence, ICTC 2018; 2018. p. 31–6. October 17-192018.
[12] Hou ES, Ansari N, Ren H. A genetic algorithm for multiprocessor scheduling. IEEE Trans Parallel Distrib Syst 1994;5(2):113–20.
[13] Santana E, Hotz G. arXiv preprint. 2016.
[14] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda,A survey of autonomous driving: common practices and emerging technologies, arXiv preprint arXiv: 1906.05113, 2019.
[15] Topcuoglu H, Hariri S, Wu MY. Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans Parall Distrib Syst 2002;13 (3):260–74.
[16] van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double q-learning. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence; 2016. p. 2094–100. February 12-172016.
[17] van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double q-learning. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence; 2016. p. 2094–100. February 12-172016.
[18] H. Zhao, Y. Zhang, P. Meng, H. Shi, L.E. Li, T. Lou, and J. Zhao, Towards safety-aware computing system design in autonomous vehicles, arXiv preprint arXiv: 1905.08453, 2019.
[19] Grondman I, Busoniu L, Lopes GAD, Babuska R. Asurvey of actor-critic reinforcement learning: Standard and natural policy gradients. IEEE Trans Syst Man Cybern Part C (Appl Rev 2012;42(6):1291–307.
[20] Redmon J, Farhadi A. Yolo9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. p. 7263–71.
[21] Wu J, Xu X, Zhang P, Liu C. Anovelmulti-agent reinforcement learning approach for job scheduling in grid computing. Future Gener Comput Syst 2011;27(5): 430–9.
[22] Cavigelli L, Benini L. Origami: A 803-gop/s/w convolutional network accelerator. IEEE Trans Circuits Syst Video Techn 2017;27(11):2461–75.
[23] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and A. Nallanathan, Deep reinforcement learning based dynamic trajectory control for uav-assisted mobile edge computing, arXiv preprint arXiv:1911.03887, 2019.
[24] Gaafar M, Shaghaghi M, Adve RS, Ding Z. Reinforcement learning for cognitive radar task scheduling. In: 2019 53rd Asilomar Conference on Signals, Systems, and Computers. IEEE; 2019. p. 1653–7.
[25] Song M, Hu Y, Chen H, Li T. Toward spervasive and user satisfactory CNN across GPU microarchitectures. In: 2017 IEEE International Symposium on High Performance Computer Architecture. HPCA; 2017. February 4-8, 2017, 2017, pp. 1-12.

Yuqiong Qi is currently a lecturer in Nursing College Jiangsu Vocational College of Medicine. She graduated from Soochow University. Her research interests include nursing education and clinical nursing.

Yang Hu is currently a lecturer in Nursing College Jiangsu Vocational College of Medicine. She graduated from Soochow University. Her research interests include nursing education and clinical nursing. She has published 6 nursing papers.

Haibin Wu is currently aassociate professor in Nursing College Jiangsu Vocational College of Medicine. She graduated from Nanjing Medical University. Her research interests include nursing education and clinical nursing. She has published 13 nursing papers

Shen Li is a associate professor in the School of Chinese studies in Xi'an International Studies University, China.His research interests include language teaching, Contrastive linguistics, more than 30 papers published and 2 books published.

Xiaochun Ye She graduated from Nanjing Medical University. Her research interests include nursing education and clinical nursing. Her research interests include nursing education and clinical nursing.

Dongrui Fan He research interests include nursing education and clinical nursing. He has published 12 nursing papers.her research on imge processing, IoT, artificial Intelligent .