AI in SAGIN: Building Deep Learning Service-Oriented Space-Air-Ground Integrated Networks

AI in SAGIN: Building Deep Learning Service-Oriented Space-Air-Ground Integrated Networks

He Li, *Member, IEEE,* Kaoru Ota, *Member, IEEE,* Mianxiong Dong, *Member, IEEE*

***Abstract*—In next-generation mobile communications, space-air-ground integrated networks (SAGINs) is an emerging infrastructure in future wireless access networks. Since artificial intelligence (AI) applications become more and more important, it is essential to build a deep learning service-oriented SAGINs. In this article, we present a hierarchical intelligent computing structure focusing on processing deep learning tasks in future SAGINs. An optimization strategy is also proposed to improve the quality-of-service (QoS) of deep learning tasks in the proposed structure. We test our work in small testbed and simulations. The evaluation results show that the proposed work outperforms other offloading strategies in a SAGIN environment.**

***Index Terms*—AGINsAGINsS, Deep Learning, AI, Service-Oriented Networking.**

I. Introduction

THE space-air-ground integrated network will play the most important role in building next-generation mobile networks [1]. Companies begin to construct several large satellite communication networks with low earth orbit (LEO) and very low earth orbit (VLEO) satellite in the space. Many researchers proposed different hierarchical network architectures with unmanned aerial vehicles (UAVs) in different attitudes in the area. Meanwhile, the 5th generation mobile communication (5G) brings millimeter-wave communications requiring a higher density of base stations than before [2]. The challenges and opportunities from space, air, and the ground will promote the development of SAGINs in the next decade.

One critical challenge is to support mobile AI applications in SAGIN architecture. Most recent deep learning-based mobile AI applications are implemented with the cloud-centric structure. Most deep learning tasks are finished in the cloud, and mobile devices are only applied for the data collection. In the SAGIN structure, the collected data from mobile devices will be transferred between ground, air, and space, which leads to very long response time. In the IoT scenarios, because of the much higher density of devices and a much larger amount of collected data, the quality of service (QoS) of AI applications will worsen [3].

Intelligent edge computing is an opportunity to improve the QoS of AI applications in mobile communications by offloading some of AI computing to the edge devices. However, since edge devices' performance is strict for offloading a small part of deep learning tasks, most applications still need to transfer collected data to the cloud through mobile networks.

One solution for improving the efficiency of the intelligent edge computing is applying high-performance edge devices for offloading, which is very difficult due to the limitation of device size and power supply capacity. Another method is to deploy different devices in the hierarchical structure of the mobile communication. Usually, devices in the base station have larger space and higher power supply than the edge devices. Therefore, it is possible to deploy devices with higher performance in the layer nearer to the backbone to offload more complex deep learning tasks.

It is also possible to apply the hierarchical offloading structure into the SAGIN for improving the QoS of AI applications. This article presents the multilayer offloading structure in space, air, and ground to support deep learning tasks in SAGINs. Different offloading devices with different performance are deployed in the SAGIN layers to offload different deep learning tasks. We also design an offloading strategy to improve the QoS of AI applications. In performance evaluation, we apply a small hierarchical AI computing testbed and extensive simulations to test the proposed structure and the offloading strategy, respectively.

The main contributions of this paper are summarized as follows.

- We investigate the issue of providing AI applications in the future SAGIN environment. To the best of our knowledge, this article is the first one focusing on offloading deep learning tasks in SAGINs.
- A hierarchical offloading structure is proposed to solve the issue in executing deep learning tasks in SAGINs. We also design an optimization strategy to improve the QoS of AI applications in the proposed structure.
- We develop a small hierarchical offloading testbed for the performance evaluation. Also, the optimization strategy is evaluated by extensive simulations.

The remainder of this paper can be outlined as follows. Section II introduces task offloading in SAGIN environments and hierarchical offloading structure in edge computing. Section III introduces the details of the hierarchical offloading structure in SAGINs. The optimization strategy in the proposed hierarchical structure is described in Section Section IV. Section V presents the proposed structure's evaluation results and the optimization strategy through a small testbed and extensive simulations, respectively. Finally, the conclusions and future work are drawn in Section VI.

H. Li, K. Ota and M. Dong are with the Department of Sciences and Informatics, Muroran Institute of Technology, Muroran, Hokkaido, Japan. E-mail: {heli, ota, mxdong}@mmm.muroran-it.ac.jp.

II. Related Work

In this section, we first discuss the task offloading in SAGIN and then introduce the hierarchical structure for offloading deep learning tasks in edge computing.

A. Task Offloading in SAGIN

Task offloading is the central issue in edge or fog computing, which is to offload different tasks from the cloud to the edge or fog devices. Task offloading usually aims to optimize two goals, smaller network traffic, and lower network latency, due to the limited bandwidth and long transmission distance from edge to the cloud. In mobile networks, since wireless communications bring higher latency with limited capacity, it is very necessary to offload a part of computing in the edge devices. A SAGIN environment has much higher latency than general mobile networks, which means task offloading becomes more critical than before.

The research on the space-air-ground integrated network is becoming a hotspot in recent years. Liu et al. surveyed about this trinity system which including the physical layer characteristics & spectrum allocation, mobility management & traffic offloading, existed architectures as well as challenges & research directions in the near future [4]. Cheng et al. developed a simulation platform to provide support for carrying out experiments and performance evaluation in space, aerial, and terrestrial networks. With their effort, researchers in this field can more easily put ideas and designs into practice [5].

Mobile edge computing, as a distributed computing solution with good scalability, is widely recognized to play a role in various cross-cutting areas. For space-air-ground integrated networks, edge computing can further improve and optimize the coordination between tiers. Zhang et al. studied the application scenarios of heterogeneous IoT in which mobile edge computing could handle the demands of multiple services from the household, medical to urban transportation [6]. Zhou et al. focused on the air-ground part while applying edge computing in assisting resource-hungry & computation-intensive applications [7]. Bekkouche et al. proposed scalable solutions on the resource allocation in the air tier. In their work, by minimizing the necessary traveled distance, each UAV can cover a longer range [8]. Callegaro et al. also considered using edge computing to enhance the work capacity of UAVs. They focused on reducing the constraint from the payload and designed a flexible autonomous airborne system [9]. Mei et al. came up with the idea of achieving task offloading between UAVs and ground base stations [10]. Wu et al. paid attention to the situation of non-uniform heterogeneous cellular networks in which edge computing can further enhance the performance of UAV-mounted offloading [11]. For the part of satellite communication, many state-of-the-art pieces of research applying mobile edge computing. Wang et al. studied the low earth orbit satellite network and proposed the concept of edge computing satellites. In their work, different accessing planes and resource requirements of terminals can be satisfied through fine-grained resource management as well as dynamic scheduling [12].

B. Hierarchical Offloading for Deep Learning

Task offloading shows good efficiency in providing most data-intensive services in mobile networks. However, it is difficult to offload compute-intensive tasks, especially deep learning tasks, to the edge due to the limited performance of general edge devices. An efficient and feasible methodology is hierarchical offloading that offloads tasks in different layers in mobile networks. For example, in an Internet of Things (IoT) environment, IoT devices connected to IoT gateways while IoT gateways are linked by the access network. Most access networks also have the hierarchical structure from the access points or base stations to the backbone. Therefore, it is possible to deploy different hardware in different layers of the mobile networks for offloading deep learning tasks.

There are two categories of hierarchical offloading methodologies for deep learning. The first is offloading different deep learning tasks in different layers of the mobile networks [13]. Usually, the edge devices have the lowest performance in executing deep learning tasks while the cloud has the highest capacity. Different deep learning tasks need different computing resources. An image recognition task is executable in an Arduino device, while a real-time video object detection task usually needs a commercial desktop processor. In task offloading, the simple tasks are offloaded to the devices near to the edge while complex tasks are offloaded near to the cloud or executed in the cloud directly.

The second methodology is offloading deep learning tasks according to the hierarchical structure of the neural networks [14]. As the fundamental computing systems, the artificial neural networks for deep learning have a multilayer hierarchical structure. Each layer receives output data of the predecessor layer and generates intermediate data to the successor. According to the neural network structure, the offloading strategy splits each neural network into different parts according to the computing requirement and the intermediate data size of each layer. Then, each part is assigned to the appropriate layer of the mobile networks.

However, the above two methodologies have different pros. and cons. The first methodology deploys complex tasks near to the cloud, which means the collected raw data will be almost transferred from the edge to the backbone with high network traffic and latency. The second methodology is designed for deep neural networks, and only parts of network structures work correctly after the split. Meanwhile, users need to develop specific programs for adaption with different mobile networks and hardware. As a result, in this article, we focus on the integration of the two methodologies in the optimization strategy and mainly apply the first methodology with better adaptability.

III. Hierarchical Offloading in SAGIN

In this section, we first investigate the scenario in supporting AI applications in SAGIN and then present the details of the hierarchical offloading structure.

A. AI applications in SAGIN

The scenario of providing AI applications in SAGIN is very important for the design of the hierarchical offloading

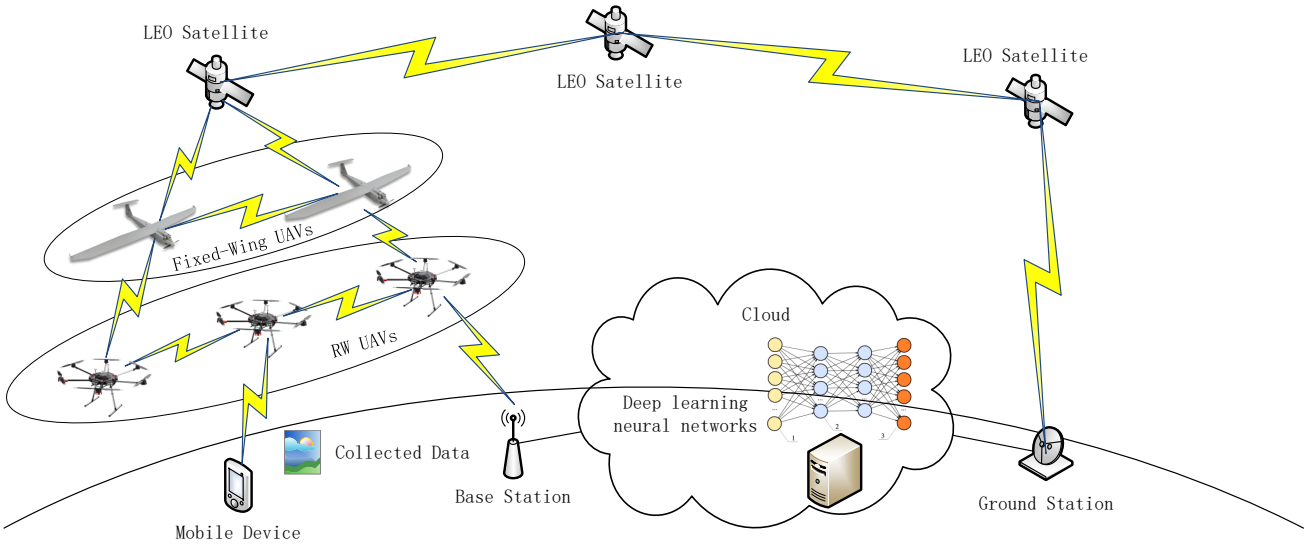


Fig. 1. Scenario of AI applications in a space-air-ground integrated network

structure. As shown in Fig. 1, we use an example to illustrate the scenario after a SAGIN user requires an AI application. In the scenario, there are six layers from the mobile device to the cloud, including the mobile device layer, base station (BS) layer, rotary-wing (RW) UAV layer, fixed-wing (FW) UAV layer, LEO satellite layer, ground station layer and the cloud layer.

The mobile device layer consists of mobile devices or IoT devices that collect data for further processing. In a digital image processing task, the collected data usually refers to digital images or videos, which have a large size in transferring. Since most mobile devices have limited communication power, most mobile networks need to deploy access points or base stations at low altitude. In the scenario, the air communication provides network access for mobile devices.

Base stations are also able to offload tasks from connected devices, including mobile devices and RW UAVs. Base stations have stable power supply and spare space to hold high-performance computing and storage hardware. However, since base stations are sparsely distributed in SAGIN covered areas, the BS layer plays an alternative role in offloading deep learning tasks.

The air communications include the FW UAV layer and RW UAV layer. FW UAVs have a large load capacity to carry large antennas for the communications between space and air. Since FW UAVs with low agility need to fly in high altitude for obstacle avoidance, RW UAVs at low altitude are applied to build communications between mobile devices and FW UAVs. Due to the load limitation, RW UAVs usually have small antennas with small transmit power to cover a small area of mobile devices.

The LEO satellite layer provides global communications between UAVs and ground stations. Since the number of ground stations is usually limited, only small parts of satellites communicate to ground stations in a short time. Therefore, a typical SAGIN builds direct links between satellites and transfers data from air to ground in multiple hops.

The ground station layer is applied to connect the LEO satellite layer and the cloud layer. A ground station has a very large antenna for providing very large communication bandwidth between satellites and ground communications.

Based on the illustrated scenario, we describe the procedures in processing deep learning tasks without offloading. The service provider first deploys the deep learning processing program in the cloud. When a user begins to use the AI application, the mobile device collects the required data and sends the collected data to the RW UAV layer. Then, the collected data will be transferred through the FW UAV layer, LEO satellite layer, and ground station layer to the cloud after several hops. After receiving the collected data, the deep learning task process the data and output the result. The result will also be transferred through the entire SAGIN. The transferring distance of both collected data and output results is very long, resulting in significant latency.

B. Hierarchical Offloading

From the above description of the scenario for providing AI applications in SAGIN, the goal of the hierarchical offloading structure is to reduce the distance between data collection and computing. Therefore, we propose the hierarchical offloading structure shown in Fig. 2.

In the hierarchical structure, we apply task offloading in five layers, the mobile device layer, BS layer, RW UAV layer, FW UAV layer, and LEO satellite layer. Because of different space, power supply, and other limitations, different layers have different hardware for offloading deep learning tasks. In the mobile device layer, a system on a chip (SoC) in each mobile device has the major processor for processing deep learning tasks. Most modern SoCs have enough performance to finish the inference parts in some simple deep learning tasks such as

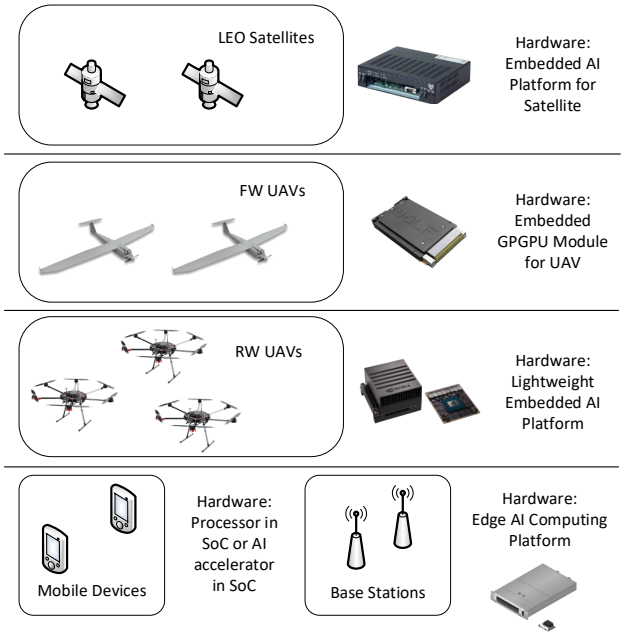


Fig. 2. Hierarchical task offloading for deep learning in SAGIN

object recognition in images or voice recognition. Meanwhile, some SoC manufacturers begin to integrate AI accelerators in commercial SoCs for better performance in supporting AI applications, which improve the offloading capability of mobile devices in task offloading.

Lightweight UAVs in the RW UAV layer usually have several kilograms payload during the flying. It is possible to a lightweight embedded AI platform, e.g., NVIDIA Jetson series, with hundreds of grams on the RW UAV for processing deep learning tasks. A typical lightweight embedded AI platform usually has the more than one trillion floating operations per second (TFLOPS) performance, which is near to an entry-level desktop general-purpose computing on the graphics processing unit (GPGPU). It is possible to offload the deepest learning tasks in the inference phase to the lightweight AI platform in the RW UAV layer.

FW UAVs usually have a very large payload with the long flight time. The payload space and the power supply are also much ampler than RW UAVs. Some products are embedding one or more desktop GPGPUs in a payload module for UAVs. The most powerful commercial OpenVTX GPGPU module compatible with FW UAVs has two NVIDIA TU104 GPGPUs with a performance of more than 40 TFLOPS. The embedded GPGPU module can offload almost all deep learning tasks even in the training phase with the mainstream computing hardware.

In the LEO satellite layer, the payload is very limited because of the expensive space launch cost. From the SpaceX company, the launch cost of the LEO satellite payload is near to 2700 dollars per kilogram, which is still much more expensive than the AI computing hardware. Meanwhile, equipment in the space usually is exposed to severe cosmic rays, which requiring robust chips with a specific design. Therefore, the LEO satellite can only carry a very light and robust device for offloading minimal parts of deep learning tasks.

As a result, we present a hierarchical offloading methodology based on the above structure. The methodology offloads tasks from the cloud to each layer of the SAGIN network according to the task parameters and QoS requirements. However, in each layer of SAGINs, devices have different performance for processing deep learning tasks and network performance for data transferring. It is necessary to find an efficient way to offload tasks into each layer of devices to guarantee the QoS requirements. In the rest of this article, we state the offloading problem in SAGINs and present a strategy to optimize the task offloading the hierarchical structure.

IV. Optimization Strategy

We first compare the differences between the five layers in the hierarchical offloading, including offloading capability, power supply, and network parameters. As shown in Table I, we list offloading parameters of five layers in the hierarchical structure. From the comparison, there are three issues as follows in the design of the offloading strategy.

First, the offloading capacity is very limited in the mobile layer, RW UAV layer, and LEO satellite layer, while FW UAVs provide the largest capacity for offloading deep learning tasks. However, the number of FW UAVs is much less than RW UAVs and mobile devices, which means it is better to offload large tasks in the FW UAV layer.

Second, the battery lifetime limited the service capacity of the mobile device layer and RW UAV layer. In hierarchical offloading, only a part of battery capacity is available for offloading deep learning tasks.

Third, the network performance of the FW UAV layer and LEO satellite layer is much worse than the lower two layers. The offloading strategy should put more deep learning tasks in the lower layers for shorter service response time.

Therefore, according to the above three issues, we formulate the optimization problem then present the offloading strategy. In AI applications, the response time is the most important QoS index of service providing. We use the average service response time as the main object of the offloading strategy.

The problem of deep learning task offloading in SAGINs: Given a set of deep learning tasks and a SAGIN, the deep learning task offloading problem attempts to schedule the deep learning tasks to five network layers in the SAGIN such that the average service response time is minimized.

According to the problem statement, we present the offloading optimization strategy to solve the offloading problem as the following steps.

Step 1: Offloading tasks in local mobile devices first until the capacity runs out.

Step 2: If a connected base station exists, offloading tasks to the base station until the base station's capacity runs out.

Step 3: Estimating the service response time of each remaining task in the cloud.

Step 4: Estimating each remaining task's service response time after offloading in the RW UAV, FW UAV, and Satellite layers.

TABLE I
COMPARISON OF FIVE LAYERS IN HIERARCHICAL OFFLOADING

Layer	Offloading Capacity	Power Supply	Uplink Bandwidth	Uplink Latency
Mobile Device	10~1000 GFLOPS	Battery (<10 Watts)	1~5 Gbps	1 ms
Base Station	1~20 TFLOPS	AC Power (10~500 Watts)	100 Gbps	10 ms
RW UAV	1~10 TFLOPS	Battery (10~30 Watts)	10 Gbps	10 ms
FW UAV	20~40 TFLOPS	Petrol Engine (100~1000 Watts)	1 Gbps	30 ms
LEO Satellite	100 GFLOPS	Solar Power (10~20 Watts)	1 Gbps	30~100 ms

TABLE II
HARDWARE IN THE TESTBED

Hardware	Operating System	Performance	Bandwidth	Layer	Height	Amount
Raspberry Pi 4	Raspbian Buster	13.5 GFLOPS	1 Gbps	Mobile Device	0 m	2
Jetson Nano	Ubuntu 18.04	472 GFLOPS	1 Gbps	FW UAV (DJI M100)	100 m	2
Jetson TX2	Ubuntu 18.04	1331 GFLOPS	1 Gbps	FW UAV (DJI M210)	100 m	2

Step 5: Calculate each remaining task's reduced service response time with offloading in the RW UAV, FW UAV, and Satellite layers.
Step 6: Calculating the quotient of expected reduced service response time divided by the required computing operations.
Step 7: Sorting remaining tasks by the above quotients in a descending sequence.
Step 8: Offloading all tasks in the sequence until the capacity of each layer is full.

The latency between nodes in the same layer is much lower than uploading latency from the lower to the upper layer in the above strategy. Therefore, we choose layers as units to offload tasks in the SAGIN with a time complexity of $O(n^2)$ where n is the number of tasks.

V. PERFORMANCE EVALUATION

In this section, we take the performance evaluation in two ways, experiments in a small testbed and extensive simulations.

A. Small Experiment

We build a small testbed with several embedded AI computing platforms and mobile devices list in Table 1. We apply two DJI Matrice 210 Series V2 (M210) and two DJI Matrice 100 (M100) drones to load Jetson Nano and TX2 at the height of 100 meters. In all experiments, we take an object detection task in the testbed. We record the total service response time, including the processing time and network latency. Since the network latency is very small in a 5 GHz Wi-Fi (IEEE 802.11ac) network environment, the network latency mainly comes from the data transferring time.

We apply a trained SSD-mobilenet V2 model as the algorithm for object detection in a high definition (1920×1080@30p) video files with 20 seconds. The video is resized down to 300×300 pixels before inputting to the model. We test the entire service response time from the Raspberry Pi to the Jetson TX2. Each node will forward the data to the next one when the task is not offloaded. We also test the testbed's scalability by recording the response time of simultaneously processing 2, 3, and 4 videos.

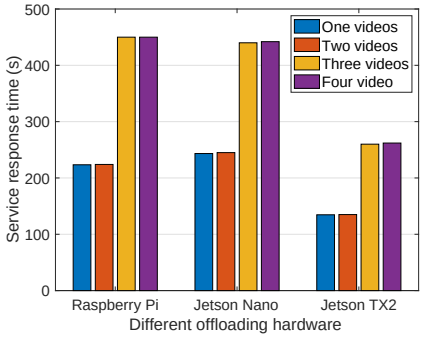
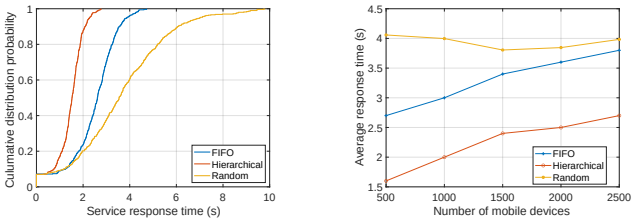


Fig. 3. Service response time with different offloading hardware

From the results shown in Fig. 3, offloading the deep learning task to the Jetson TX2 brings the lowest service response time. After offloading the task to Jetson Nano, the service response time is two times langer than TX2. Because of the time for data transfer, the service response time after offloading tasks to Jetson Nano nodes is similar to the processing time on Raspberry Pi nodes. Therefore, the hardware performance will be a significant part of processing deep learning tasks with enough network performance. However, if the data size becomes larger with limited network performance, the deep learning task offload efficiency will be more critical.

B. Simulation Result

We also take extensive simulations to evaluate the performance of the optimization algorithm. All simulations are developed by the NetworkX 2.5 Python library. In all simulations, we apply 500 mobile devices, 2 base stations, 10 RW UAVs, 5 FW UAVs, and three satellites for building the SAGIN. Considering a satellite can stably cover a given point no more than 100 seconds, we assume all deep learning tasks can be processed in [0.1,1] seconds, and the entire simulation time period is 50 seconds. In each second, every mobile device will send a deep learning task with a probability of 0.05 until there is not enough time for processing tasks. The device capacity of each layer is set as shown in Table I. The collected data size of each task is uniformly distributed in [10, 1000]



(a) Cumulative distribution distribution of average response time

(b) Average response time with different number of mobile devices

Fig. 4. Average response time with hierarchical offloading

MB. The required computing capacity is uniformly distributed in [10, 2000] GLOPS.

We first test the cumulative distribution probability of each mobile device's average service response time. As shown in Fig. 4(a), the average response time with the optimization offloading strategy is much shorter than the FIFO strategy and random strategy for all mobile devices. The longest average response time with the random strategy is worse than 10 seconds. If we use the service satisfaction ratio as another measurement of QoS and set the required average response time is less then 2 seconds, the optimization offloading strategy can have near to 80% satisfaction ratio while the other two methods only archive 20%.

Since the service capacity usually limits the QoS, we also test the average response time with the different number of mobile devices. We added 500 mobile devices in each step and recorded the average response time in 5 steps. As shown in Fig. 4(b), the optimization strategy in the hierarchical offloading has the shortest average response time, even with 2500 mobile devices. The average response time with the FIFO strategy is near to 1.5 times longer than the optimization strategy. When the number of mobile devices becomes 2500, the FIFO strategy performs similarly with the random strategy.

VI. Conclusion and Future Work

In this article, the hierarchical structure shows good efficiency for processing deep learning tasks in SAGINs. Due to the inherent latency for transferring data from the edge to the cloud, the novel offloading structure will improve the QoS of providing AI applications. Meanwhile, a well designed offloading strategy is also very important in the hierarchical offloading structure. In the future, we plan to build a more powerful testbed with more embedded AI computing platforms. Meanwhile, a new strategy for a long period offloading will be studied with the consideration of the movement of satellites, UAVs, and mobile devices.

Acknowledgment

This work is partially supported by JSPS KAKENHI Grant Numbers JP19K20250, JP20K11784, JP20H04174, JP22K11989, Leading Initiative for Excellent Young Researchers (LEADER), MEXT, Japan, and JST, PRESTO Grant Number JPMJPR21P3, Japan. Mianxiong Dong is the corresponding author.

References

[1] E. Yanmaz, R. Kuschnig, and C. Bettstetter, "Achieving air-ground communications in 802.11 networks with three-dimensional aerial mobility," in *2013 Proceedings IEEE INFOCOM*. IEEE, apr 2013.

[2] J. Xu, K. Ota, and M. Dong, "Energy efficient hybrid edge caching scheme for tactile internet in 5g," *IEEE Transactions on Green Communications and Networking*, vol. 3, no. 2, pp. 483–493, jun 2019.

[3] R. Morabito, V. Cozzolino, A. Y. Ding, N. Beijar, and J. Ott, "Consolidate IoT edge computing with lightweight virtualization," *IEEE Network*, vol. 32, no. 1, pp. 102–111, jan 2018.

[4] J. Liu, Y. Shi, Z. M. Fadlullah, and N. Kato, "Space-air-ground integrated network: A survey," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 2714–2741, Fourthquarter 2018.

[5] N. Cheng, W. Xu, W. Shi, Y. Zhou, N. Lu, H. Zhou, and X. Shen, "Air-ground integrated mobile edge networks: Architecture, challenges, and opportunities," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 26–32, August 2018.

[6] Y. Zhang, Y. Wu, H. Moustafa, D. H. K. Tsang, A. Leon-Garcia, and U. Javaid, "Multi-access mobile edge computing for heterogeneous iot," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 12–13, August 2018.

[7] Z. Zhou, J. Feng, L. Tan, Y. He, and J. Gong, "An air-ground integration approach for mobile edge computing in iot," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 40–47, August 2018.

[8] O. Bekkouche, T. Taleb, M. Bagaa, and K. Samdanis, "Edge cloud resource-aware flight planning for unmanned aerial vehicles," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, April 2019, pp. 1–7.

[9] D. Callegaro, S. Baidya, and M. Levorato, "A measurement study on edge computing for autonomous uavs," in *Proceedings of the ACM SIGCOMM 2019 Workshop on Mobile AirGround Edge Computing, Systems, Networks, and Applications*, ser. MAGESys'19. New York, NY, USA: Association for Computing Machinery, 2019, p. 29–35. [Online]. Available: https://doi.org/10.1145/3341568.3342109

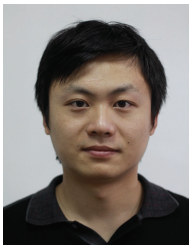
[10] H. Mei, K. Wang, D. Zhou, and K. Yang, "Joint trajectory-task-cache optimization in uav-enabled mobile edge networks for cyber-physical system," *IEEE Access*, vol. 7, pp. 156 476–156 488, 2019.

[11] H. Wu, Z. Wei, Y. Hou, N. Zhang, and X. Tao, "Cell-edge user offloading via flying uav in non-uniform heterogeneous cellular networks," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2020.

[12] F. Wang, D. Jiang, S. Qi, C. Qiao, and H. Song, "Fine-grained resource management for edge computing satellite networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*, Dec 2019, pp. 1–6.

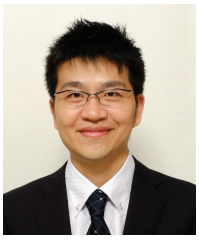
[13] L. Li, K. Ota, and M. Dong, "Deep learning for smart industry: Efficient manufacture inspection system with fog computing," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4665–4673, oct 2018.

[14] J. Zhou, Y. Wang, K. Ota, and M. Dong, "AAIoT: Accelerating artificial intelligence in IoT systems," *IEEE Wireless Communications Letters*, vol. 8, no. 3, pp. 825–828, jun 2019.



He Li received the B.S., M.S. degrees from Huazhong University of Science and Technology in 2007 and 2009, respectively, and the Ph.D. degree from The University of Aizu in 2015. He is currently an Assistant Professor and MEXT Excellent Young Researcher with Muroran Institute of Technology, Japan.

Kaoru Ota received M.S. degree from Oklahoma State University in 2008 and B.S. and Ph.D. degrees from The University of Aizu in 2006 and 2012, respectively. She is an Associate Professor and MEXT Excellent Young Researcher with Muroran Institute of Technology, Clarivate Analytics 2019 Highly Cited Researcher and JST-Presto researcher.



Mianxiong Dong received his B.S., M.S., and Ph.D. in computer science and engineering from the University of Aizu. He is a professor in the Department of Sciences and Informatics at Muroran Institute of Technology, Japan where he serves as the Vice President. He is Clarivate Analytics 2019 Highly Cited Researcher.

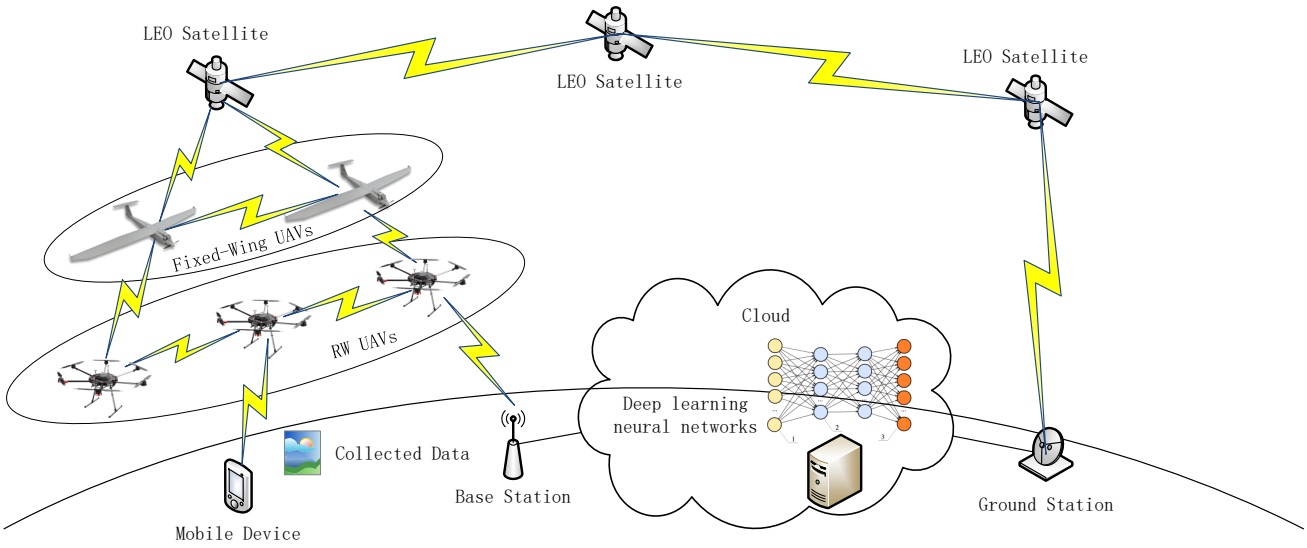


Fig. 1. Scenario of AI applications in a space-air-ground integrated network

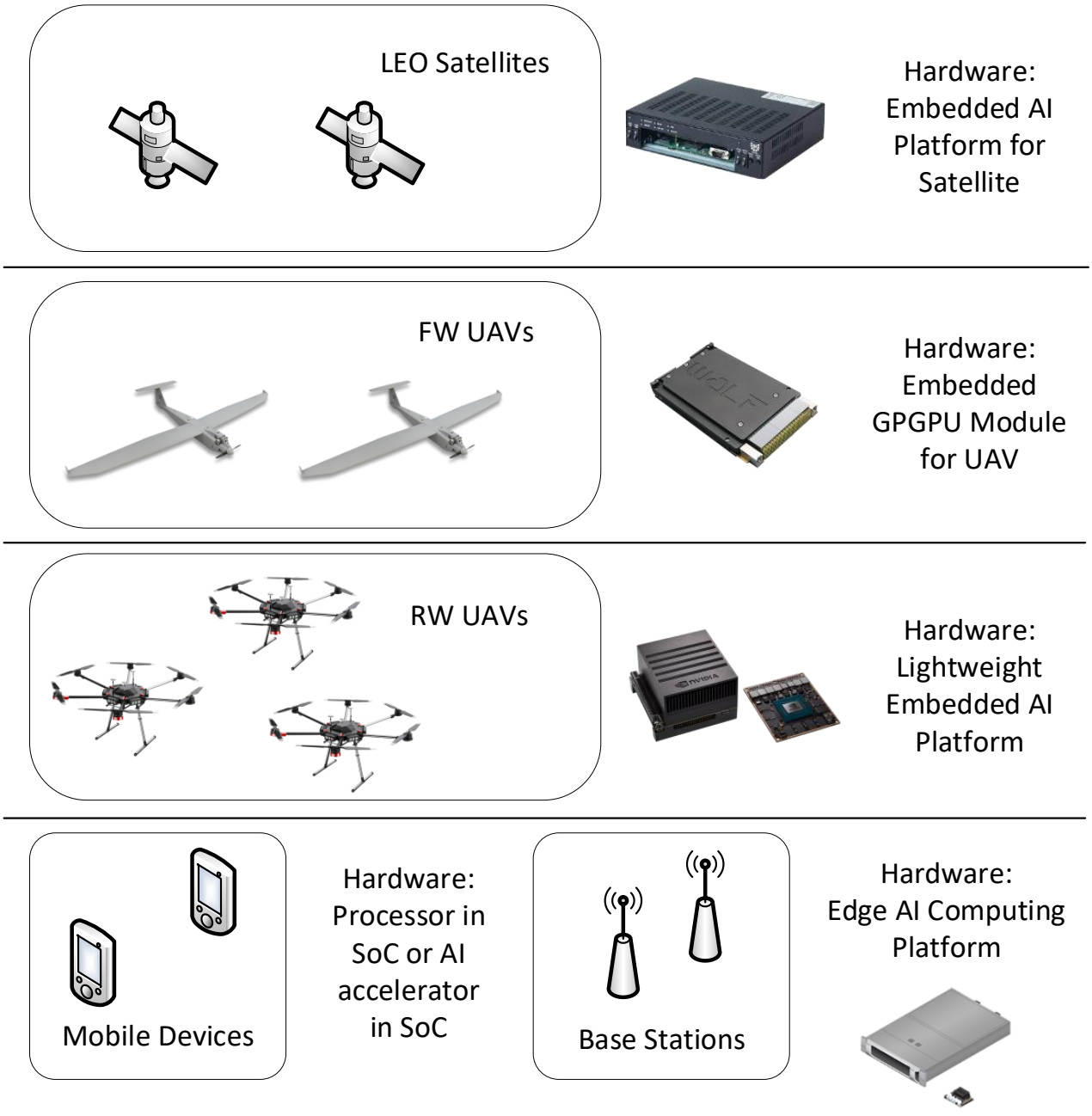


Fig. 2. Hierarchical task offloading for deep learning in SAGIN

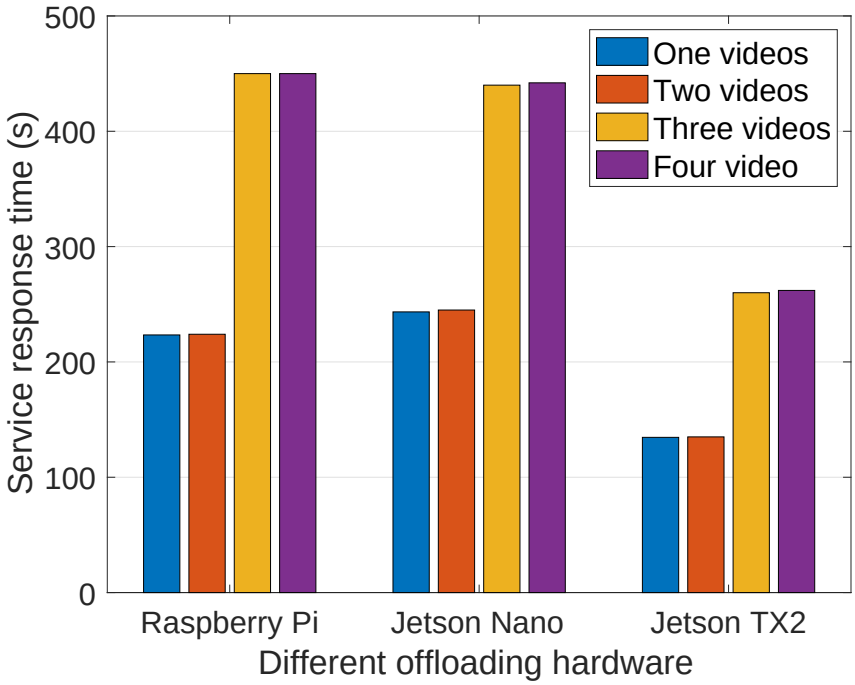
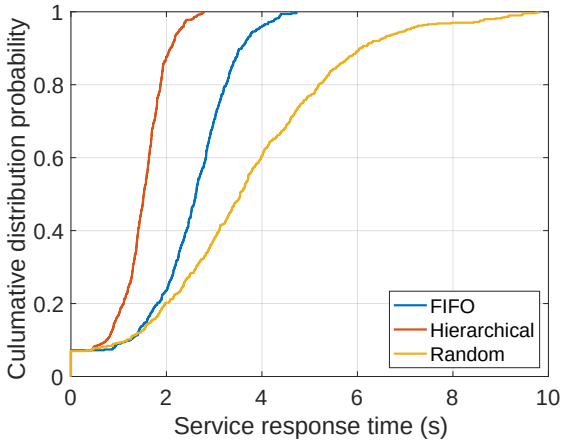
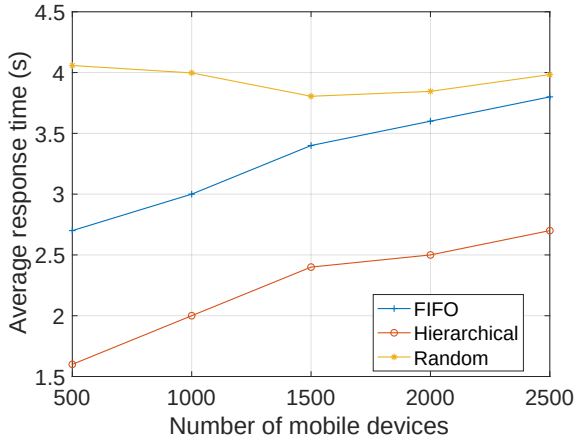


Fig. 3. Service response time with different offloading hardware



(a) Cumulative distribution distribution of average response time



(b) Average response time with different number of mobile devices

Fig. 4. Average response time with hierarchical offloading