International Journal of Pure and Applied Mathematics
Volume 119 No. 10 2018, 531-543
ISSN: 1311-8080 (printed version); ISSN: 1314-3395 (on-line version)
url: http://www.ijpam.eu
Special Issue

ijpam.eu

Flexible Deep Learning in Edge Computing for IoT

[1]Sneha Sureddy, [2]K. Rashmi, [3]R. Gayathri and [4]Archana S. Nadhan

[1]Department of CSE,
GITAM, Bengaluru, Karnataka.
sneha.sureddy@gmail.com

[2]Department of CSE, GITAM,
Bengaluru, Karnataka.
rashmikjul11@gmail.com

[3]Department of CSE, GITAM,
Bengaluru.
gayurams@gmail.com

[4]Department of CSE,
GITAM, Bengaluru.
archana.snathan@gitam.edu

Abstract

Deep learning is a promising approach for extracting accurate information from raw sensor data from IoT devices deployed in complex environments. Because of its multilayer structure, deep learning is also appropriate for the edge computing environment. Traditional edge computing models have rigid characteristics. Flexible edge computing architecture solves rigidity in IoT edge computing. Proposed model combines deep learning into edge computing and flexible edge computing architecture using multiple agents. Since existing edge nodes have limited processing capability, we also design a novel offloading strategy to optimize the performance of IoT deep learning applications with edge computing. FEC architecture is a flexible and advanced IoT system model characterized by environment adaptation ability and user orientation ability. In the performance evaluation, we test the performance of executing deep learning tasks in FEC architecture for edge computing environment. The evaluation results show that our method outperforms other optimization solutions on deep learning for IoT.

Key Words: IoT, deep learning, FEC, edge computing.

1. Introduction

The Internet of Things (IoT) has become an important field because it can provide services based on real time contextual information. IoT devices generate a huge amount of data due to the large number of end devices. In order to extract insights from huge volume of IoT data in real time, processing needs to happen near to the end device where the data is generated (the edge). However, most of the machine learning and AI algorithms require a significant amount of processing capacity which may not always be available at the edge. We present an architecture that is flexible for processing at edge/cloud and provides a good balance between the cost of processing data and benefits at the central location vs edge.

Cloud computing platform has witnessed significant transitions in its overall usage, size, computational ability, and underlying technology. The enhanced flexibility and reliability attributes offered by the cloud computing paradigm have led to its widespread popularity among academia and industry. However, with the emergence of the IoT, the need for real time data storage, access, and processing at the cloud has grown manifold. Moreover, the big data generated by the connected devices would be on the order of zetta bytes in the near future. Hence, the relaying on cloud infrastructure for storing such huge data may create network bottlenecks in the future. Additionally, this would lead to latency issues which affects the overall quality of service (QoS) for various applications in IoT. In order to tackle the above limitations of the cloud platform, the concept of edge computing is introduced. It is popularly known as the cloud close to the ground, as it provides computational and processing facilities at the edge of the network.

Transferring huge data from end device to central base station is an important challenge. To overcome this problem edge computing. There are two layers available in IOT network, which are edge layer and cloud layer to connect IoT devices and the cloud services. The edge layer comprises of IoT devices, an IoT gateway and network access points in local area networks. The cloud layer includes the Internet connections and cloud servers. Edge layer generates and transmits data to cloud layer. Cloud layer process received data. Edge computing means the processing is performed in the edge layer instead of the cloud layer. In the edge computing environment, only the intermediate data or results need to be transferred from the end devices to the cloud service, hence the pressure on the network is relieved with less transferring data. Edge computing is very suitable for the application in which the size of intermediate data is smaller than the input data. Therefore, edge computing is efficient for deep learning tasks, since the size of extracted features is reduced by the filters in deep learning network layers.

Deep learning is becoming an emerging technology for IoT applications and systems. The most important benefit of deep learning task over machine learning is better performance with large data because many IoT applications generates huge amount of data for processing. Another benefit is that deep learning can extract new features automatically for different problems. Traditional machine learning algorithms depends on accuracy of the features identified and extracted by processing multimedia information. Since it can precisely learn high-level features such as human faces and voices, deep learning can improve the efficiency of processing multimedia information. Deep learning takes much less time to inference information than traditional machine learning methods. Lane et al. [1] proposed new acceleration engines, such as DeepEar and DeepX, which supports different deep learning applications in the latest mobile systems on chips (SoCs). From the experimental results, mobile IoT devices with high-spec SoCs can support part of the learning process. Introducing deep learning into more IoT applications is another important challenge [2]. The efficiency of deep learning for IoT has been evaluated in many different IoT applications. For example, some works focus on the applications in wearable IoT devices deployed in dynamic and complex environments that often confuse the traditional machine learning methods. Bhattacharya et al. [3] proposed a new model for deep learning for wearable IoT devices that improves the accuracy of audio recognition tasks.

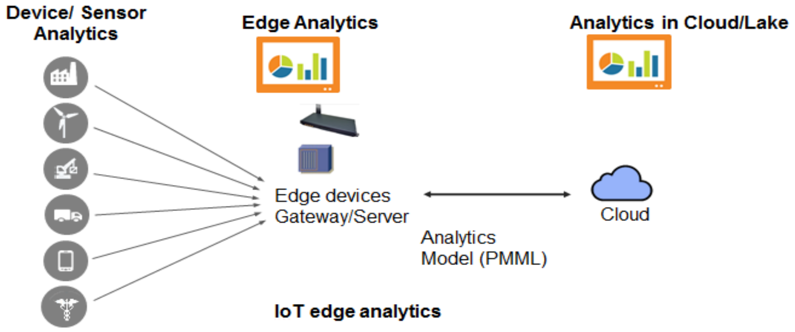


Figure 1: IoT Edge Computing

2. Related Work

Developers of Edge Network Cloud Sim simulation tool performed study [4] on high throughput Edge network, performed improvements to “CloudSim” framework [5] by reorganizing algorithms and results to evaluate service times and failures, opposing to energy consumption, that is more reliable measurement for Cloud computing. Furthermore, introduced customization of data centers and service chains allows to evaluate its orchestration. Experimental investigation is performed on network topology consisting of 13 nodes and 15 edges, evaluating only service times. Additionally, comparison of results of doubling bandwidth and delays is performed.

Related investigation is performed by simulating intelligent surveillance system [6], [7], where “iFogSim” toolkit is used. Comparison between Cloud and Edge-

Fog computing is performed in terms of average control loop latency, execution time. Five different physical network topology configurations are evaluated, revealing 15% energy and 25s execution time optimization by pushing surveillance system controls to Edge rather than performing computations in Cloud. Edge computing integration together with blockchain research [8] outlines possibility to use peer-to-peer distributed blockchain technique beyond transactions verification. Three tier model is used for research, which reveals bottleneck in load that can be processed in real time.

For smart city and fourth industrial revolution applications [9], “iSapiens” [10] Edge Computing based platform is proposed. Main advantages of platform is complete network virtualization, layered structure, followed by anomaly detection and aggregation agents, complete obscure heterogeneity of physical devices. As summarized in [11], one of the main challenges incurred by edge computing is the task offloading, i.e., how to properly partition the tasks into sub-tasks and efficiently allocate them to nearby edge nodes. A widely studied theory to provide a guide line for task offloading is called the Divisible Load Theory (DLT) [12].

This work first divides the divisible tasks into multiple sub-tasks of distinct sizes and then independently processes them with computing nodes for a minimum completion time. Following the basic idea of DLT, the authors in [12] and [13] proposed an optimal partition method and an optimal load allocation method, respectively. Specially, [13] deduces a general result that if all processors finish the processing at the same time, the minimum completion time is achieved. Further, two problems are investigated in [14], [15], namely the minimum completion time with all the processors working and the minimum number of processors required to meet an application’s deadline. The aforementioned researches mainly explore how to partition tasks and allocates sub-tasks to edge nodes to finish a task before the deadline. However, for some tasks, it is essential to obtain the best processing results within the deadline not just considering the minimum completion time, such as preprocessing video tasks with a higher quality in video surveillance systems, which can benefit the performance of video analysis in further steps.

Most existing deep learning applications (e.g., speech recognition) still need to be cloud-assisted. Alsheikh et al. [16] proposed a framework to combine deep learning algorithms and Apache Spark for IoT data analytics. The inference phase is executed on mobile devices, while Apache Spark is deployed in cloud servers for supporting data training. This two-layer design is very similar to edge computing, which shows that it is possible to offload processing tasks from the cloud. Video tasks can be divided into sub-tasks and preprocessed by edge nodes (i.e., mobile devices with redundant resources) [17]. It is widely observed that the number of edge nodes is more than that of video sub-tasks in realistic scenarios. Most existing deep learning applications (e.g., speech recognition) still need to be cloud-assisted. Alsheikh et al. [19] proposed a framework to combine deep learning algorithms and Apache Spark for IoT data analytics. Flexible architecture is presented in [22] provides the multiple agents to service in edge computing. [23] provides the deep learning layer service to manage the task at the edge computing to enhance the performance of the IoT.

3. Proposed Model

Usually, IoT devices generate huge amounts of data and transfer that data to the cloud for further processing. These data include structured data, such as temperature, vibration or multimedia information, such as video, images, and sounds. Edge computing is proposed to move computing ability from centralized cloud servers to edge nodes near the user end. In this manner when applying EC to an IoT framework, the roles of edge and cloud should be established in order to optimize IoT framework with task assignment between edge and cloud varied dynamically according to the resources and environment situation of the edge. However, the present EC lacks flexibility, that is, it has rigid architecture. Its presumed ill effects include deterioration of system performance, lower scalability, prevention of functional development, and unsatisfied user requests. Proposed model uses the flexible framework to provide optimized assignment of the task to the edge layer and cloud layer which reduces the rigidity. Also proposed model uses the deep learning in edge computing for flexible architecture using multiple agents, which increases the performance of the system. Combination of the deep learning for edge computing and flexible architecture for edge computing and produce better performance compare to the stand-alone flexible architecture.

Deep Learning and Edge Computing

Edge computing is proposed to move computing ability from centralized cloud servers to edge nodes near the user end. Edge computing provides two major improvements to the existing cloud computing. The first one is that the nodes in the edge layer can preprocess large amounts of data before transferring them to the central servers in the cloud. The second one is that the cloud resources are optimized by enabling edge nodes with computing ability [20]. Liu et al. [21] proposed a deep-learning based food recognition application by edge-computing-based service model and it shows that edge computing improves the performance of deep learning applications significantly by reducing response time and lowering energy consumption. IoT devices generate large amounts of data and transfer data to the cloud server, such as include multimedia data and scalar data. Processing multimedia data is much complex than processing scalar data. Traditional multimedia information processing technologies needs complex computations, which are not appropriate for IoT services. Since the deep learning technology greatly improves the efficiency of processing multimedia information, and becomes important research topic. Video sensing is an important IoT application, which integrates image processing and computer vision in IoT networks. Recognize objects from low-quality video data recorded by IoT devices is still challenging task**.** Since deep learning shows very promising accuracy in video recognition, we consider it as a typical IoT application with deep learning. Deep learning in IoT is performed as layerd processing in distributed manner between edge and cloud. The task is divided into sub task and assigned to layers in the deep learning process. The layers are distributed between edge and cloud. Edge layers are lower layers and cloud layers are called higher layers. IoT devices generates data and given to deep learning layers in edge computing, lower layers process the data and produce intermediate data or results and it is transferred to cloud, where higher layers process the intermediate data and produce results. A problem is how to divide each deep learning network. Usually, the size of the intermediate data generated by the higher layers is smaller than that generated by the lower layers. Deploying more layers into edge nodes can reduce more network traffic because of transmitting less data. However, the capacity of edge node is limited compared to cloud servers. It is impossible to process infinite tasks in edge nodes. We can only deploy part of the deep learning network into edge nodes. This creates the rigid architecture for assigning task to edge and cloud.

Flexible Edge Computing

Flexible edge computing FEC and Its Architecture — Optimization of task assignment between edge and cloud in IoT system is a fundamentally important research theme. We propose Flexible EC (FEC) architecture to resolve the problems caused by the rigid characteristics of present EC mentioned in the previous section. The typical three-layer IoT architecture fundamentally comprises three layers, a perception layer that recognizes a matter and collects its state and environment data, a network layer that collects data obtained at the perception layer, and an application layer that provides various services using data collected according to user needs for each field of application. In the typical five-layer IoT architecture, common functions are taken out from each layer of the three-layer IoT architecture into a separate layer, and a business layer is appended.

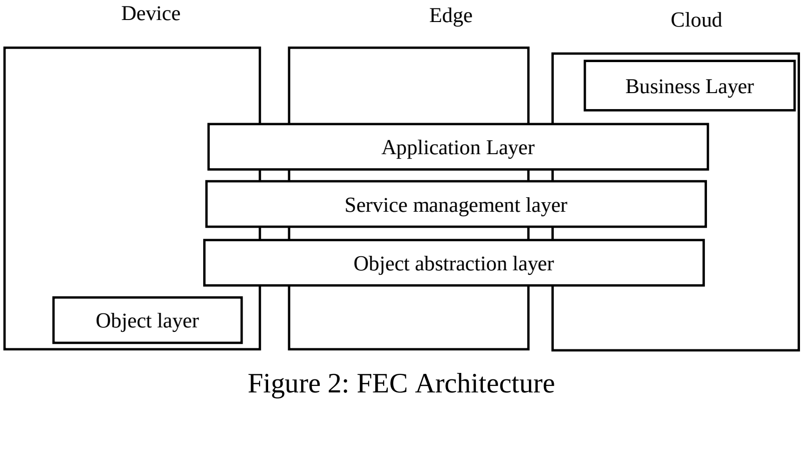


Figure 2: FEC Architecture

Figure 2 shows the five-layer IoT architecture proposed by Al-Fuqaha et al. [18]. Five-layer IoT architecture, in which the role of the service management layer is enhanced by giving its task to manage the device, edge, and cloud integratively. The object abstraction layer is extended to manage all components of EC from sensors and devices to a network, this makes the foundation for enhancement of the service management layer.

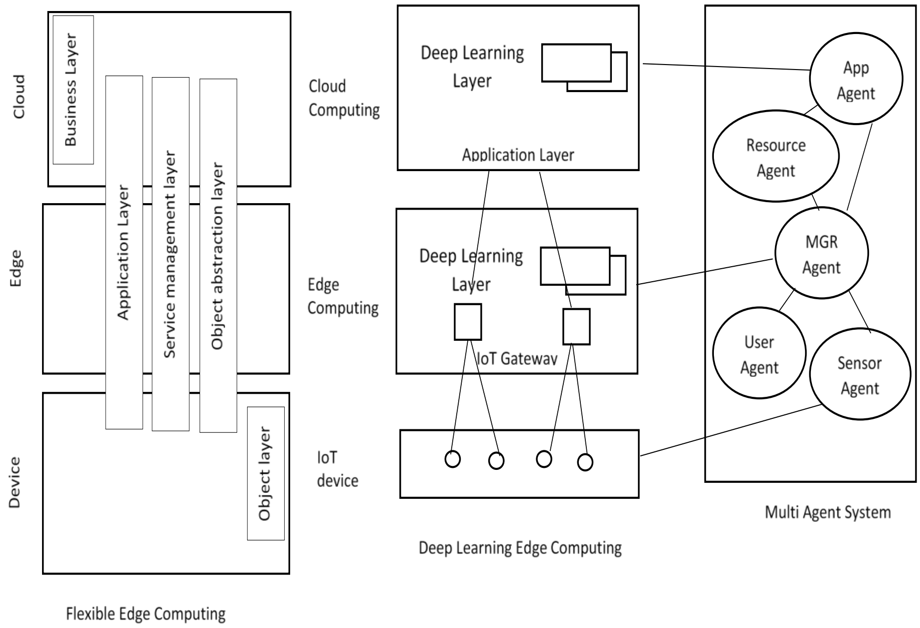


Figure 3: FEC with Deep Learning Edge Computing

Furthermore, the application layer is extended to the device domain by reflecting the actual situation in which application services are provided. By these extensions, FEC architecture can have advantages including improvement of overall system performance, scalability enhancement, quick response to changes in IoT components, and so on. To realize FEC architecture, we adopt two system abilities: environment adaptation ability and user orientation ability. Environment adaptation ability autonomously determines the allocation of processing to an edge and a cloud according to the quality and quantity of work and their fluctuation, whereas user orientation ability provides services suitable for each user in real time by reflecting detailed information collected by IoT, such as a user’s behavior, intention, and preference by autonomous control. We think that these two abilities are missing in the current EC and that it makes the EC rigid, a disadvantage mentioned in the previous section.

Multi-Agent based FEC with Deep Learning

Figure 3 depicts the configuration of the multi agent-based FEC architecture currently being examined as one candidate architecture for implementing FEC [8]. The traditional IoT architecture is shown in the center of the figure; the

FEC architecture is drawn on the left side, and the multi agent-based implementation of the FEC architecture is shown on the right side as a logical mechanism to control. A FEC platform with deep learning layer is physically distributed and deployed in each component of EC in practice. It operates as a logical control mechanism overall.

Deep learning layer is attached to each layer in the traditional architecture. The platform is configured as a set of software agents that control each FEC architecture component (i.e., a multi-agent system). Multiple agents may be deployed in the same device or hardware; thus, a software infrastructure called an agent platform should be installed in each device to separate or coordinate their needs. Broken lines shows the relation between each component and an agent controlling it. An agent is practically deployed on each device or hardware carrying a control object. The environment adaptation ability of FEC architecture is implemented by the agent framework and dedicated protocols among the agents. Each agent is furnished with a protocol for constituting an organization based on the privity of contract. This implements dynamic configuration and reconfiguration of a virtual organization according to the environment change. Because a multi agent system generally takes a flat structure, the task assignment of components according to conditions is possible irrespective of the EC architecture. Consequently, the environment adaptation ability can be implemented. On the other hand, the user orientation ability of FEC architecture is implemented as a user agent. The user agent, formed by abstracting a user, grasps the preference and service request of the user in charge through interaction with the user. This user agent, acting as a mediator between a user and the system, allows embedding a user into the system, so the user orientation ability is realized based on the human-in-the-loop concept. Each agent is adapted with deep learning layers.

FEC mainly includes the following five types of agents:

User agent: It converts a request from a user into a format that is interpretable by other agents.

MGR agent: It manages sensors and apps that are dynamically generated by a user request and a contract concluded by them.

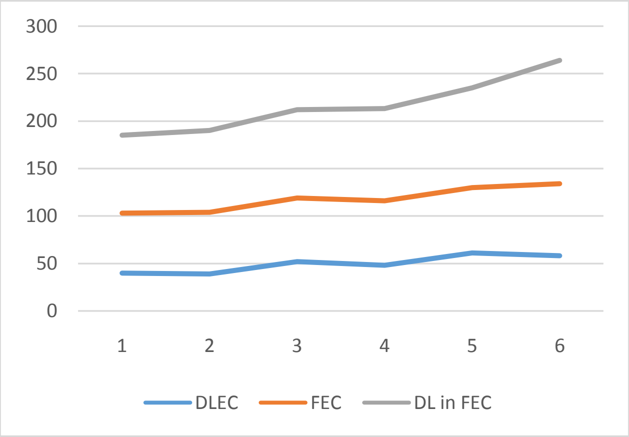
Sensor agent: It functions as a sensor device registered by a user. It concludes a contract with the agents of another user according to a predetermined provision policy and conducts provision of sensor data based on the contractual coverage.

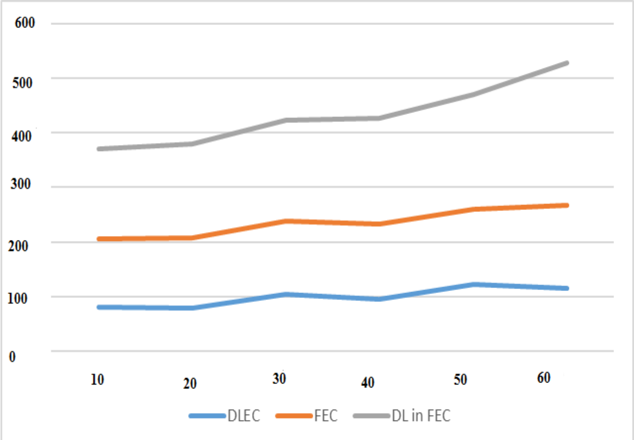
App agent: It functions as an application constructed by a user. It concludes a contract with the agents of another user according to a user request, and conducts acquisition, processing, and provision of sensor data circulated based on the contractual coverage.

Resource agent: It grasps and manages the conditions of the computer resources and network resources of each device and instrument, and conducts information provision to other agents upon request.

4. Performance Evaluation

In this section, we first describe the experiment settings and then discuss the performance evaluation result. In the experiments, we have two environments, one for collecting data from deep learning tasks and another for simulations. For executing deep learning applications, we use a workstation equipped with an Intel Core i7 7770 CPU and NVIDIA Geforce GTX 1080 graphic card. We use Caffe as the FEC framework and define 10 different FEC networks. We execute 10 FEC tasks with different FEC networks and record the number of operations and intermediate data generated in each deep learning layer. We use Python 2.7 and networkx to develop the simulator and use the reduced ratio of the intermediate data from executing FEC tasks. In the simulations, we set the number of deep learning tasks to 1000. The service capability of each edge server is set to 290 Gflops according to NVIDIA Tegra K1. We set the number of edge servers in the network from 20 to 90. The input data size of each task is set from 100 kB to 1 MB. The layer number of all FEC networks is set from 5 to 10. The bandwidth of each edge server is uniformly distributed from 10 Mb/s to 1 Gb/s. Proposed model performs deep learning in FEC architecture using multiple agent for edge computing. We input a random sequence of 1000 tasks to the edge network, and this algorithms deploy tasks into edge servers. The output is compared with deep learning in edge computing and Flexible architecture edge computing. Proposed model combines the deep learning and flexible architecture and produce better performance using multiple agents. The following results shows that proposed scheme produce high throughput compared to other methodology for different situations.





5. Future Work and Conclusion

This article presents a proposal of FEC architecture as a flexible system to perform edge computing using deep learning in IoT. Combination of these two models that are deep learning and flexible edge computing significantly improves the performance of the system and optimize the task assignment between edge layer and cloud layer. Proposed model outperforms the other methodology. Our future tasks include the preparation of many applications and examination of the contributions of this approach through verification experiments.

References

[1] Lane N.D., Georgiev P., Qendro L., Deepear: Robust Smartphone Audio Sensing in Unconstrained Acoustic Environments Using Deep Learning, Proc. ACM Int'l. Joint Conf. Pervasive and Ubiquitous Computing (2015), 283–94.

[2] Li L., Eyes in the Dark: Distributed Scene Understanding for Disaster Management, IEEE Trans. Parallel Distrib. Systems (2017).

[3] Bhattacharya S., Lane N.D., Sparsification and Separation of Deep Learning Layers for Constrained Resource Inference on Wearables, Proc. 14th ACM Conf. Embedded Network Sensor Systems CD-ROM, ser. SenSys 16 (2016), 176–89.

[4] Seufert M., Kwam B.K., Wamser F., Tran-Gia P., Edge network cloud sim: Placement of service chains in edge clouds using network cloud sim, In NetSoft (2017), 1–6.

[5] Calheiros R.N., Ranjan R., Beloglazov A., De Rose C.A., Buyya R., Cloud sim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning

algorithms, Software: Practice and experience 41(1) (2011), 23–50.

[6] Gupta H., Vahid Dastjerdi A., Ghosh S.K., Buyya R., iFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments, Software: Practice and Experience 47(9) (2017), 1275–1296.

[7] Alsmirat M.A., Jararweh Y., Obaidat I., Gupta B.B., Internet of surveillance: a cloud supported large-scale wireless surveillance system, The Journal of Supercomputing 73(3) (2017), 973–992.

[8] Stanciu A., Blockchain based distributed control system for edge computing, IEEE International Conference on Control Systems and Computer Science (CSCS) (2017), 667–671.

[9] Sapienza M., Guardo E., Cavallo M., La Torre G., Leombruno G., Tomarchio O., Solving critical events through mobile edge computing: An approach for smart cities, IEEE International Conference on Smart Computing (SMARTCOMP) (2016), 1–5.

[10] Cicirelli F., Guerrieri A., Spezzano G., Vinci A., An edge based platform for dynamic smart city applications, Future Generation Computer Systems (2017), 106–118.

[11] Varghese B., Wang N., Barbhuiya S., Kilpatrick P., Nikolopoulos D.S., Challenges and opportunities in edge computing, IEEE International Conference on Smart Cloud (Smart Cloud) (2016), 20–26.

[12] Ghose D., Robertazzi T., Foreword (special issue of cluster computing on divisible load scheduling), Cluster Computing 6(1) (2003), 5–5.

[13] Kyong Y., Robertazzi T., Greedy signature processing with arbitrary location distributions: a divisible load framework, IEEE Transactions on Aerospace and Electronic Systems 48(4) (2012), 3027–3041.

[14] Bharadwaj V., Ghose D., Robertazzi T., Divisible load theory: a new paradigm for load scheduling in distributed systems, Cluster Computing 6(1) (2003), 7–17.

[15] Lin X., Lu Y., Deogun J., Goddard S., Real-time divisible load scheduling for cluster computing, In IEEE Real Time and Embedded Technology and Applications Symposium (2007), 303–314.

[16] Mamat A., Lu Y., Deogun J., Goddard S., An efficient algorithm for real-time divisible load scheduling, In IEEE Real-Time and

Embedded Technology and Applications Symposium (2010), 323–332.

[17] Eriksson E., Dn G., Fodor V., Predictive distributed visual analysis for video in wireless sensor networks, IEEE Transactions on Mobile Computing 15(7) (2016), 1743–1756.

[18] Al-Fuqaha A., Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications, IEEE Commun. Surveys & Tutorials 17(4) (2015), 2347–76.

[19] Alsheikh M.A., Mobile Big Data Analytics Using Deep Learning and Apache Spark, IEEE Network 30(3) (2016), 22–29.

[20] Tran T.X., Collaborative Mobile Edge Computing in 5G Networks: New Paradigms, Scenarios, and Challenges, IEEE Commun. Mag., 55(4) (2017), 54–61.

[21] Liu C., A New Deep Learning-Based Food Recognition System for Dietary Assessment on an Edge Computing Service Infrastructure, IEEE Trans. Services Computing (2008).

[22] Takuo Suganuma, Takuma Oide, Shinji Kitagami, Kenji Sugawara, Norio Shiratori, Multi agent-Based Flexible Edge Computing Architecture for IoT, Edge Computing for the Internet of Things (2018).

[23] He Li, Kaoru Ota, Mianxiong Dong, Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing, Edge computing for the internet of things (2018).