

Performance Evaluation of Partial Offloading under Various Scenarios in Mobile Edge Computing

SeokBeom OH*, SooJeong LEE*, Yong-Geun HONG**, Gyuyeol KONG***, Hyun-Kook KAHNG****

* Program in Converging Technology systems and Standardization, Korea University, South Korea

** Department of Artificial Intelligence & Convergence, Daejeon University, South Korea

*** Institute for Hokmah General Education, Chongshin University, South Korea

**** Department of Electronics and Information Engineering, Korea University, South Korea

isb6655@korea.ac.kr, ngenius@korea.ac.kr, yghong@dku.ac.kr, gykong@csu.ac.kr, kahng@korea.ac.kr

The problem of offloading policy is addressed for mobile edge computing (MEC) in this paper. We proposed a deep learning-based partial offloading method to reduce user equipment's energy consumption and service delay. The proposed method consists of two deep neural networks (DNNs) to find the best partitioning of a single task and their offloading policy, respectively. Multiclass classification is used for the selection of partitioning and offloading policies. For partitioning selection, the DNN was learned through the ratio of task size instead of the actual task size to improve the classification accuracy. The performance of the proposed method was evaluated in three scenarios which are delay-critical model (DCM), energy-critical model (ECM), and delay and energy-critical model (DECM). The simulation results show that ECM has the worse classification performance for partitioning selection than DCM and DECM, while three scenarios have similar classification performance for offloading selection. Additionally, the proposed method has more than 77% and 89% classification performances for partitioning and offloading in various scenarios, respectively.

Keywords—Mobile edge computing, computation offloading strategy, task partitioning, deep learning.

I. INTRODUCTION

Mobile edge computing (MEC) technology has emerged to reduce the service delay and energy consumption of user equipment(UE) by deploying powerful computing and storage capacity when the computing tasks are offloaded to mobile edge server (MES)[1]. In MEC, the energy consumption and service delays are greatly influenced by the offloading policy that determines which task components are executed locally or remotely at MES. The offloading strategy can be categorized into full offloading and partial offloading[2]. In partial offloading, the task is divided into several components. Then some of the components are offloaded to MES for execution, while the whole task is offloaded to MES for execution in full offloading[3].

However, it is a complicated problem to find optimal partial offloading policy-making process faster and energy-efficient because the number of possible ways of task partitioning increases exponentially with the task size[4]. Therefore, research on how the task components can be divided efficiently between UEs and MES to reduce UEs' service delay and the energy consumption.

Several computational offloading methods have been introduced to minimize UE service delay and energy consumption in MEC systems[1]–[7]. These methods[1]–[7] are compared and analyzed from the perspective of a service delay, energy consumption, task partitioning, and deep learning in [7]. Among them, the deep learning-based partial offloading method[7] consumes less energy with faster execution in MEC networks compared to the conventional methods[1]–[6]. However, this method performs multiclass classification with numerous classes by considering partitioning and offloading policy together, making classification problems complex and unreliable. The number of classes partitioning and offloading policy increase exponentially as the data size of a single task and the number of components per task increase. Considering complexity and reliability, we perform multiclass classification task with an appropriate number of classes regardless of data size.

This paper proposes a deep learning-based partial offloading method to minimize service delay and energy consumption with the reduced number of classes for partitioning and offloading policy selections. In the proposed method, two things are changed compared to the conventional method[7]. First, a deep neural network (DNN) that considers the selections of partitioning and offloading policy together is divided into two DNNs for partitioning and offloading policy selections, respectively. It could significantly reduce the combination of classes between partitioning and offloading policy. Second, to further reduce the number of classes, the class for partitioning selection is generated through a ratio of size rather than actual size. It mitigates the rapid increase in the number of classes as data size increases. By using two techniques, the proposed method performs classification tasks through significantly reduced classes, enabling reliable selections for partitioning and offloading policy with low complexity. The performance of the proposed method was evaluated in various scenarios. Performance comparison and analysis are performed through various models in environments where service delay is the dominant factor, energy consumption is the dominant factor, and both service delay and energy consumption are considered as the dominant factor.

II. SYSTEM MODEL

This section presents a method of calculating the cost function for partial offloading and deep learning model structure.

A. Cost Function

We consider the system model [7] to find the optimal offloading policy and partitioning in MEC. The detailed formulas for obtaining cost and system parameters can be found in [7]. $f_{cl}(c_i)$ indicates the local cost when c_i executes locally in the UE and $f_{co}(c_i)$ indicates the remote cost when c_i executes remotely on MES, where c_i is the i th components of a single task. The cost function for local execution $f_{cl}(c_i)$ can be calculated as:

$$f_{cl}(c_i) = \delta_1 \left(\frac{d_{li} + d_{di}}{d_{max}} \right) + \delta_2 \left(\frac{E_{li} + E_{di}}{E_{max}} \right), \quad (1)$$

where δ_1 and δ_2 are the weighting coefficient by which we can change the contribution and priority of delay and energy consumption in the local cost function, respectively. In the delay parameters, d_{li} is total delay for local execution of c_i , d_{di} is time delay for task-division per component, and d_{max} is the deadline time for a whole task execution. In the energy parameters, E_{li} is total energy consumption for local execution of c_i , E_{di} is energy consumption for partitioning of the task and E_{max} is maximum energy of UE's battery. Similarly, the cost function for remote execution $f_{co}(c_i)$ can be describe as:

$$f_{co}(c_i) = \delta_3 [d_{ti}(1 - e_{i-1}) + d_{ei} + d_{ri} + d_{pi}(2 - e_{i-1}) + d_{di}] + \delta_4 [E_{ti}(1 - e_{i-1}) + E_{ri} + E_{di}] + \delta_5 f_2(r_i) + \delta_6 f_3(k_i), \quad (2)$$

where δ_3 , δ_4 , δ_5 and δ_6 are the weighting coefficient by which we can change the contribution and priority of time delay, energy consumption, radio resources and computational resources in the cost function. We set the sum of all coefficients to be 1. In the delay parameters, d_{ti} , d_{ei} , d_{ri} , and d_{pi} are the delay due to transmission, execution, reception, propagation for remote execution of c_i , respectively. E_{ti} and E_{ri} are the energy consumption in the energy parameters due to transmission and reception for remote execution of c_i , respectively. We multiply $(1 - e_{i-1})$ with d_{ti} and E_{ti} if the previous component c_{i-1} was executed at MES, where e_i is indicator that means 0 when c_i executes locally and 1 when c_i executes remotely at MES. It means the next component c_i is available at MES and does not need to be retransmitted. Similarly, $d_{pi}(2 - e_{i-1})$ gives propagation delay only for reception. $f_2(r_i)$ and $f_3(k_i)$ give the cost due to used CPU cores and subcarriers, respectively, where r_i represents the number of CPU cores used in the processing of c_i and k_i represents the available subcarriers of c_i . We change the delay and energy priorities to identify and analyze how the weighting coefficient effect on delay and energy changes when offloading is performed. In order to consider various delays and energy priorities, we evaluate the energy consumption and service delay of partial offloading by setting weighting coefficients according to three scenarios.

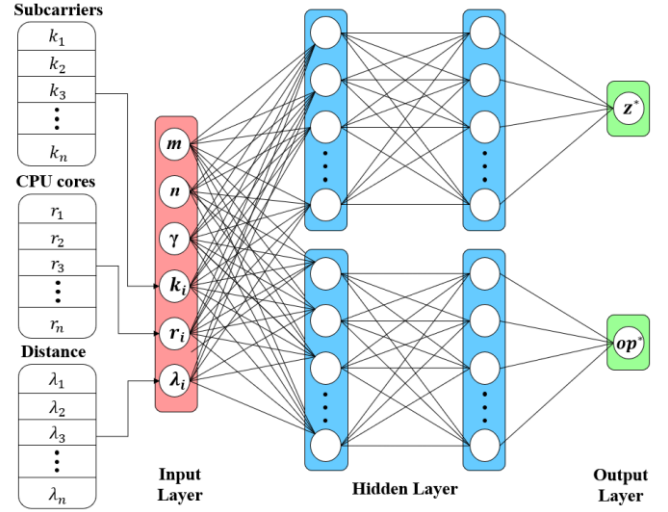


Figure 1. The proposed deep neural network structures

B. Deep Learning Model

The complexity of conventional model [7] determining optimal partitioning and offloading policy increases exponentially. To reduce this very large computational complexity, we use a supervised learning approach. The DNN performs offloading and partitioning selections based on the given input values. The proposed technique consists of two DNNs to find the best partitioning of a single task and their offloading policy.

In the conventional DNN structure in [7], it could exponentially increase the combination of classes between partitioning and offloading policy. To solve this problem, we divided single DNN into two DNNs, one DNN for best option for task division and the other offloading decision for each component. For partitioning selection, the class is generated through a ratio of size rather than actual size. It mitigates the rapid increase in the number of classes as data size increases. To generate a dataset for training deep neural network models, we selected and saved the option and input data with minimum cost. We randomly generated task size, number of components, division resolution, number of subcarriers, available computing resources of MES, distance between UE and MES, frequency of UE, transmitting power of UE to consider simulations in various situations. All DNN models consist of three types of layers. It consists of an input layer, a hidden layer, and an output layer. The input layer consists of the information about the size of task (m), number of components (n), division resolution (γ), and network status (k), available computing resources of MES (r), and distance from MES (λ). The hidden layer consists of two layers for each deep neural network model. Each hidden layer consists of 128 neurons. Each of two output Layer provides different information, one output layer for optimal offloading policy(op^*) and another for optimal partitioning(z^*). Overall DNN structures are shown in Figure 1.

TABLE 1. WEIGHT FOR EACH MODEL

WEIGHT	DCM	ECM	DECM
δ_1	0.8	0.2	0.5
δ_2	0.2	0.8	0.5
δ_3	0.6	0.2	0.4
δ_4	0.2	0.6	0.4
δ_5	0.1	0.1	0.1
δ_6	0.1	0.1	0.1

TABLE 2. TEST ACCURACY FOR EACH MODEL

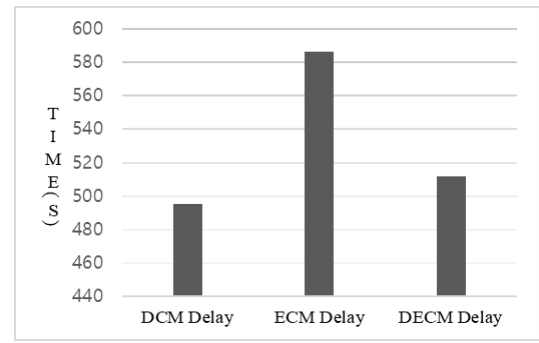
Model	Test Accuracy	
	Partitioning	Offloading Policy
DCM	83.03%	89.14%
ECM	77.34%	89.53%
DECM	82.74%	89.7%

III. SIMULATION RESULT

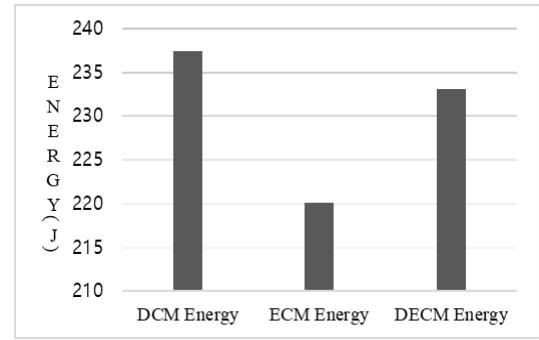
The overall simulation environment and parameters are shown in [7]. There are some changes in our simulation compared to [7]. We fixed the number of components per task to 6 and fixed the division resolution to 200MB. Therefore, 21 neurons in the input layer; a neuron for task size, a neuron for number of components per task, a neuron for division resolution, six neurons for random distances during mobility of UEs in the execution of each component, six neurons for subcarriers during the execution of each component, and six neurons for CPU cores assigned to each component. Similarly, six neurons for offloading policy and six neurons for partitioning are reserved in each output layer. The ReLU and Softmax activation function are used for each hidden layer and the output layer, respectively. We generated 200,000 simulation data for deep learning. The data is randomly divided into training, validation, and test data by 70%, 20%, and 10%, respectively.

Table 1 shows the weight coefficient of the three models. Considering application-specific requirements, we set the weight values according to the characteristics of each model. We considered three different models, which are delay-critical model (DCM), energy-critical model (ECM), and delay and energy-critical model (DECM).

Table 2 shows the classification accuracies of partitioning and offloading policy selections. The classification performances of partitioning are 77% to 83%, and the classification performances of offloading policy are 89% for all models. The ECM has the lowest accuracy in partitioning classification because the amount of energy that depends on the input of ECM is low.



(a) Service delay comparison for each model



(b) Energy consumption comparison for each model

Figure 2. Service delay and energy consumption for each model

Figure 2 shows the results of the effect of delay and energy weighting on actual delay and energy by DCM model, ECM model, and DECM model, respectively. We performed a comparison between delay and energy using the result of partitioning prediction and offloading policy prediction of each model.

Figure 2(a) shows the comparison results from the perspective of delays in each model. As we adjusted the weight coefficient, the DCM model had the shortest delay, and the DECM model had the second short delay. The ECM model showed the most prolonged delay because the delay does not matter in the ECM model. Figure 2(b) shows the comparison results from the perspective of energies in each model. Similarly, the ECM model had the lowest energy consumption, followed by the DECM model with the second-lowest energy consumption, and the DCM model showed the highest energy consumption.

IV. CONCLUSIONS

In this paper, we proposed a deep learning-based partial offloading method in MEC systems for various scenarios. Two DNNs were employed for the selections of the partitioning of a single task and their offloading policy to solve the multiclass classification problems, respectively. For partitioning selection, the ratio of task size instead of the actual task size was considered as label data. The performance of the proposed method was evaluated in various scenarios. The simulation results showed that the proposed method has more than 77% and 89% classification performances for partitioning and offloading selection in various scenarios,

respectively. Applying appropriate weights depending on the specific requirements of the application will contribute to reducing latency and energy. Our future work is to find a way to further increase accuracy by applying a local search algorithm to the partitioning classification selection.

ACKNOWLEDGMENT

This research was supported by the Korea Evaluation Institute of Industrial Technology project, "Standardization of Networking of Everything Architecture and Protocols (20002532)" and supported partially by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (2021-0-00188, Open-source development and standardization for AI enabled IoT platforms and interworking)

REFERENCES

- [1] Z. Liao, J. Peng, B. Xiong, and J. Huang, "Adaptive offloading in mobile-edge computing for ultra-dense cellular networks base on genetic algorithm," *Journal of Cloud Computing*, vol. 10, no. 15, 2021.
- [2] X. Gu, C. Ji, and G. Zhang, "Energy-optimal latency-constrained application offloading in mobile-edge computing," *Sensors* 2020, 20, 3064.
- [3] B. Jo, M.J. Piran, D. Lee, and D.Y. Suh, "Efficient computation offloading in mobile cloud computing for video streaming over 5G," *Comput. Mater. Contin.* 2019, 61, 439–463.
- [4] H. Zhang, Y. Yang, X. Huang, C. Fang, and P. Zhang, "Ultra-low latency multi-task offloading in mobile edge computing," *IEEE Access* 2021, 9, 32569–32581.
- [5] L. Ale, N. Zhang, X. Fang, X. Chen, S. Wu, and L. Li, "Delay-aware and energy-efficient computation offloading in mobile edge computing using deep reinforcement learning," *IEEE Trans. Cogn. Commun. Netw.* 2021, Early Access.
- [6] A. Shakarami, A. Shahidinejad, and M. Ghobaei-Arani, "An autonomous computation offloading strategy in Mobile Edge Computing: A deep learning-based hybrid approach," *J. Netw. Comput. Appl.* 2021, 178, 102974.
- [7] Z.H. Abbas, Z. Ali, G. Abbas, L. Jiao, M. Bilal, D.-Y. Suh, and M.J. Piran, "Computational Offloading in Mobile Edge with Comprehensive and Energy Efficient Cost Function: A Deep Learning Approach," *Sensors* 2021, 21, 3523.

research interests include artificial intelligence and Edge computing, IoT



Yong-Geun Hong is an Assistant Professor of the Department of Artificial Intelligence & Convergence, Daejeon University, Daejeon, Korea. He was a director of the Electronics and Telecommunications Research Institute (ETRI) from 2001 to 2020. He was a project leader of IoT Network and Intelligent Network R&D projects in ETRI. He received his B.S., M.S. and Ph.D in computer engineering from the Kyungpook National University, Daegu, Korea. He is also working for the IoT related standardization at IETF and ITU-T. His research interests include Internet Protocol, IoT, network intelligence, and edge computing.



Gyuyeol Kong (M'19) received the B.S. and Ph.D. degrees in electrical engineering from the School of Electrical and Electronic Engineering, Yonsei University, Seoul, South Korea, in 2007 and 2014, respectively. From 2014 to 2016, he was a Senior Engineer with Samsung Electronics, Ltd., Hwaseong, South Korea. From 2016 to 2019, he was a Research Professor with the School of Electrical and Electronic Engineering, Yonsei University, Seoul, South Korea. From 2019 to 2021, he has a Postdoctoral Researcher with the Department of Signal Processing and Acoustics, Aalto University, Espoo, Finland. Since 2021, he has been an Assistant Professor with the Institute for Hokmah General Education, Chongshin University, Seoul, South Korea. His research interests include signal processing for radar, communications and storage systems, error correction codes, and machine learning.



Hyun-Kook Kahng is a Professor of Department of Electronics and Information Engineering at Korea University Sejong campus, Sejong city, Korea. He received bachelor's degree in Electronics Engineering from Korea University in 1982, master's degree in Computer Engineering from University of Michigan, Ann Arbor in 1984, and Ph.D. in Computer Networking from Georgia Institute of Technology in 1990. From 1991 to 1994, he was a Senior Research Engineer at ETRI, Korea. Since 1994, he has been a professor in Korea University. He has been standardizing in various SDO's such as ITU-T, ISO/IEC JTC1 SC6 and IETF in the field of computer networking. Since 2015, he has been the chair of JTC1.SC6. His current research interest includes, future networking technologies, especially for IoT, Blockchain.



SeokBeom Oh received his B.S. degree in electronics and information engineering, Korea University, Sejong, South Korea, in 2019. He is now a Master Student of Program in Converging Technology systems and Standardization at Korea University in South Korea. His



SooJeong Lee received his Master of Science degree in Computer engineering, Korea University, Anam, South Korea. He is now a Ph.D. student of Program in Converging Technology system and Standardization in Korea University in South Korea. He is working at KT, master of ABC consulting Div. (ABC: AI/ Bigdata/ Cloud) His research interests include Cloud and MEC.