Deep Neural Network Task Partitioning and Offloading for Mobile Edge Computing

Mingjin Gao[†§], Wenqi Cui[†], Di Gao[†], Rujing Shen[†§], Jun Li[‡], Yiqing Zhou[†§]
[†]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, CHINA
[§]University of Chinese Academy of Sciences, Beijing, CHINA
[‡]School of Electronic and Optical Engineering, Nanjing University of Science and Technology, Nanjing, CHINA
Email: gaomingjin@ict.ac.cn, shenrujing@ict.ac.cn, jun.li@njust.edu.cn

***Abstract*—The surging Deep Neural Network (DNN) based applications are becoming increasingly popular in mobile computing. However, they impose significant challenges for mobile computing, as DNN tasks lead to much more computation complexity and data volume compared with traditional tasks. To alleviate this, mobile edge computing (MEC) provides a feasible approach through task partitioning and offloading. In this paper, we investigate a DNN based MEC scheme considering multiple mobile devices and one MEC server. To facilitate task partitioning, we first develop a processing delay prediction mechanism for typical DNN tasks. To achieve the minimal processing delay as well as to release the computing burden of mobile devices, a mixed integer linear programming (MILP) based DNN task partitioning and offloading mechanism is presented. Evaluations show that our mechanism can achieve up to 90.5% and 69.5% processing delay reduction compared with MEC server only and mobile device only schemes respectively.**

***Index Terms*—Task partitioning and offloading, deep neural network, MILP, mobile edge computing**

I. Introduction

Nowadays, mobile intelligent applications such as Apple Siri and Microsoft Cortana etc., become increasingly popular in our daily life. These intelligent applications have huge demands for computing power while our mobile devices only have limited computing power. To alleviate this problem, the traditional solution is transmitting all the data from mobile devices to cloud servers for processing. However, with the evolution of the artificial intelligence discipline, Deep Neural Network (DNN) based applications have shown their power and become increasingly popular in mobile computing. Such applications invoke tremendous computation requirements and may introduce a sheer volume of data that needs to be transmitted if being processed by cloud servers. It can be found that cloud-only processing isn't necessarily the best choice for all DNN based tasks due to its large transmission delay. To accelerate the DNN task execution, there are many related studies [1–5]. To reduce the amount of the workload transmissions, in [1], authors proposed dynamic DNN design techniques for efficient workload allocation. Experiments have been comprehensively performed on several well-known DNN models and datasets. Their results show that, on an average, the proposed techniques are able to reduce the amount of transmissions by up to 17% compared to previous methods under the same accuracy requirement. In [2], authors proposed an efficient computational method, which is inspired by a computational core of fully connected neural networks, to process convolutional layers of state-of-the-art deep CNNs within strict latency requirements. Simulation results showed that the proposed accelerator can process convolutional layers of VGGNet up to 9.5 times faster than state-of-the-art accelerators reported to-date while occupying 3.5 mm^2. However, the works mentioned above accelerate the DNN task execution only by improving the DNN design, which is difficult to be widely applied. We need to design a universal computing framework which is suitable for all DNN tasks.

Mobile edge computing (MEC) has recently become quite popular, since it can reduce both computation and data transmission requirements for mobile devices [6–9]. To achieve this, MEC employs a task partitioning and then offloading fashion. Many studies have been put on the MEC scheme design in the last few years [10, 11]. In [10], authors proposed a partial offloading approach, where the tradeoff between fog node energy consumption and task processing delay is considered. They also estimated the portion to be offloaded to the available devices at the edge of the network by comparing a centralized and a distributed architecture. In [11], authors studied the efficient task offloading schemes in vehicular edge computing networks. They proposed a location-based offloading scheme to consider tradeoffs between the task completed latency and the required communication and computation resources. However, the aforementioned works only consider traditional tasks whose computation complexity and data volume are much smaller than those of DNN tasks. There are many challenges in DNN offloading. For example, how to partition a deep neural network task with multiple layers for offloading? And when there are multiple mobile devices, how to jointly partition their DNN tasks for offloading? These challenges motivate our work.

In this paper, we study a DNN task partitioning and offloading problem among multiple mobile devices and one MEC server. To solve the challenge of jointly partitioning DNN tasks for offloading, we adopt the mixed-integer linear programming (MILP) theory. MILP is a linear programming whose partial decision variables are integers. MILP has been widely applied in many wireless communication studies [12,

978-1-7281-0962-6/19/$31.00 ©2019 IEEE

13]. In [12], a MILP approach was proposed for assigning heterogeneous robot teams to the simultaneous completion of sequences of tasks, considering specific requirements such as completion deadlines. In [13], a MILP-based uninhabited combat aerial vehicles task allocation scheme was proposed to search space and evolution among populations.

The main contributions of this paper are listed as follows:

1) To facilitate DNN task partitioning, a processing delay prediction mechanism is developed for typical DNN tasks.

2) A MILP-based DNN task partitioning and offloading mechanism is proposed. The mechanism considers multiple mobile devices and one MEC server;

3) The evaluation for the mechanism is conducted. Results demonstrate that our mechanism can always achieve the minimal processing delay of DNN tasks, compared with traditional methods.

The rest of this paper is organized as follows. In Section II, we present the system model. We formulate the MILP-based deep neural network task partitioning and offloading model in Section III. In Section IV, we present performance evaluation results. Finally, we draw our conclusions in Section V.

II. System Model

In this section, we detail the system model. We first describe the DNN task partitioning and offloading model in general. Then, we introduce the delay prediction scheme, which contributes to the construction of our optimization problem.

A. DNN Task Partitioning and Offloading Model

We consider there are N mobile devices and one MEC server. Each mobile device has a DNN task to be processed. To reduce the processing delay as well as the computing burden of mobile devices, each DNN task can be partitioned and be offloaded to the server who has strong computing power. Note that the DNN being used is a trained network, which means there is no DNN training involved in our DNN task partitioning and offloading model. The DNN has M layers, such as convolutional layer, fully connected layer, pooling layer and so on. We assume that these layers are relatively independent. It means that, to process the workload of one layer, we only need the output from the former layer. Thus, we regard the workload at one layer as one subtask. As a result, we can obtain M subtasks from one original task.

Task partition means that, we partition one task into two parts, the former part will be processed locally. Then, the local output data will be transmitted to the server, which means the later part will be processed by the server. Consider the relative independence of subtasks, we can partition each DNN task between layers. To describe the partition patterns mathematically, we introduce an $N \times M$ matrix $\mathbf{X}$ whose element is x_{ij}, where x_{ij} is a $0-1$ variable. We define that $x_{ij} = 1$ when the j-th subtask of i-th mobile device is processed by the server. And $x_{ij} = 0$ means that the j-th subtask of i-th mobile device is processed locally. Thus, if $x_{i1} = \cdots = x_{ij} = 0, x_{i,j+1} = \cdots = x_{iM} = 1$, it means that we partition the i-th mobile device's DNN task between the j-th layer and the $(j+1)$-th layer. The subtasks before the j-th layer are processed locally, and the subtasks after the $(j+1)$-th layer are processed by the server.

B. Delay Prediction Scheme

Our DNN task partitioning and offloading scheme is designed to minimize the average processing delay which is defined by $\overline{T}$. To construct our optimization problem, we should first analyse related delays during the DNN task partitioning and offloading process. Then, we can build a delay prediction model for scheme design.

DNN has multi-type layers, and layers with different types have different structures. As a result, the influence factors as well as prediction model for execution delay in each layer is various. Moreover, even for the same layer type, the prediction models for execution delay on server and on mobile devices may be different. In addition, the size of output data generated from each layer is dynamically changing, which leads to the variation of data transmission delay between the mobile devices and the server.

To build the delay prediction scheme, we first obtain influence factors for each layer according to the source code of Caffe framework, such as data size of input and output, kernel size and so on. Then, we test the relationship between each influence factor and this layer's execution delay on the mobile device or on the server by variable-controlling approach. As a result, we can know the influence pattern of each factor to the execution delay, i.e., linear, quadratic etc.. Moreover, we can also select the effective factors who have significant influence, and delete useless factors for model simplification. Based on the selected factors and expected influence patterns, we use polynomial fitting to build our execution delay prediction model.

Take the convolutional layer for example. We first select the influence factors, i.e., convolution kernel size K, size of input data I, size of output data O, size of output image G, and length of each image L. Then, we repeat the execution delay experiments for over 1000 times, and obtain a great number of data for influence factors and execution delay. We separate these data into training set and test set. Specifically, the training set has 80% of the original data, and the test set has 20% of the original data. Based on the data in training set, we obtain our polynomial fitting prediction model as follows

$$T^l = (0.3G^2K^2OI + 2.6G^2K^2I + 4.8G^2O) \times 10^{-5}, \quad (1)$$

$$T^s = (3G^2K^2OI + 7.4G^2K^2I + 25.8G^2O) \times 10^{-7}, \quad (2)$$

where T^l and T^s are the local execution delay and server execution delay for this layer, respectively. Finally, we apply the prediction model in (1) and (2) to the test set. The experimental results show that, the R-square of the convolutional layer model in the test set is 99.58%, and the average absolute error is 2.78 milliseconds.

The transmission delay is given by [14]

$$T^t = \frac{O}{r}, \tag{3}$$

where r presents the uplink rate in the channel between the mobile device and the server, and O denotes the size of data transmitted from the mobile device to the server.

III. Deep Neural Network Task Partitioning and Offloading Design

In this section, we investigate the MILP-based DNN task partitioning and offloading design in detail.

Our main objective is to reduce the average processing delay $\overline{T}$ by determining partition patterns and subtasks processing schedule. We first define an $N \times M$ matrix $\mathbf{S}$ to describe start execution time of each subtask. Its element s_{ij} presents the start execution time of i-th mobile device's j-th subtask. These elements are in time sequence $s_{i1} < \cdots < s_{iM}$, $i = 1, \cdots, N$. Similarly, we define an $N \times M$ matrix $\mathbf{F}$ as finish execution time of each subtask. Its element f_{ij} presents the finish execution time of i-th mobile device's j-th subtask. And we have $f_{i1} < \cdots < f_{iM}$, $i = 1, \cdots, N$. Then, we can give the the average processing delay $\overline{T}$ as follow

$$\overline{T} = \frac{1}{N} \sum_{i=1}^{N} f_{iM}. \tag{4}$$

Intuitively, the start execution time of the first subtask is always larger than 0. It can be denoted by

$$s_{i1} \geq 0, \ i = 1, \cdots, N. \tag{5}$$

Moreover, considering that each subtask is processed after obtaining the output data produced from the former subtask, start execution time of each subtask must be larger than the finish execution time of the former subtask. It can be denoted by

$$s_{ij} \geq f_{i,j-1}, \ i = 1, \cdots, N. \tag{6}$$

For each subtask, if it is processed locally, its finish execution time should be lager than the sum of start execution time and local processing time. Otherwise, if it is processed by the server, its finish execution time should be lager than the sum of start execution time and server processing time, i.e.

$$f_{ij} \geq s_{ij} + (1 - x_{ij}) T^l_{ij} + x_{ij} T^s_{ij}, \ i = 1, \cdots, N, \tag{7}$$

where T^l_{ij} denotes the local execution time of the i-th mobile device's j-th subtask, and T^s_{ij} denotes the server execution time of the i-th mobile device's j-th subtask. T^l_{ij} and T^s_{ij} are derived from our delay prediction model in Section II-B.

Assume that we partition the i-th mobile device's DNN task between the k-th layer and the $(k+1)$-th layer, which means $\sum_{j=1}^{k} x_{ij} = 0$ and $\sum_{j=1}^{k+1} x_{ij} = 1$. Then, the start execution time of the $(k+1)$-th subtask must be larger than the sum of the k-th subtask's finish execution time and output data transmission time between the k-th layer and the $(k+1)$-th layer $T^t_{k,k+1}$, i.e.

$$s_{i,k+1} \geq f_{ik} + T^t_{k,k+1}, \ \text{if} \ \sum_{j=1}^{k} x_{ij} = 0 \ \text{and} \ \sum_{j=1}^{k+1} x_{ij} = 1, \tag{8}$$

where $T^t_{k,k+1}$ is derived from the expression (3) in Section II-B.

In addition, the server can only process one subtask at once, which means that there is no interval overlap among $[s_{ij}, f_{ij}]$ for all subtasks whose $x_{ij} = 1$. It can be denoted by

$$f_{ij} \leq s_{pq}, \ \text{if} \ x_{ij} = x_{pq} = 1 \ \text{and} \ s_{ij} < s_{pq}. \tag{9}$$

Now, we can write the optimization problem as follows

$$\min_{\mathbf{X},\mathbf{S},\mathbf{F}} \ \overline{T} \tag{10}$$

$$\text{s.t.} \ \ 0 \leq s_{i1} < \cdots < s_{iM}, \tag{11}$$

$$f_{i1} < \cdots < f_{iM}, \tag{12}$$

$$s_{ij} \geq f_{i,j-1}, \tag{13}$$

$$f_{ij} \geq s_{ij} + (1 - x_{ij}) T^l_{ij} + x_{ij} T^s_{ij}, \tag{14}$$

$$s_{i,k+1} \geq f_{ik} + T^t_{k,k+1}, \tag{15}$$

$$\text{if} \ \sum_{j=1}^{k} x_{ij} = 0 \ \text{and} \ \sum_{j=1}^{k+1} x_{ij} = 1,$$

$$f_{ij} \leq s_{pq}, \ \text{if} \ x_{ij} = x_{pq} = 1 \ \text{and} \ s_{ij} < s_{pq}. \tag{16}$$

We find that the unknown variables are either integers (i.e. $\mathbf{X}$) or real numbers (i.e., $\mathbf{S}$ and $\mathbf{F}$). Thus, the optimal solution is an MILP problem, which can be solved using standard software packages.

Algorithm 1 Algorithm for DNN task partitioning and offloading strategy

Input: N, M, K, I, O, G, L.
Output: $\mathbf{X}^*, \mathbf{S}^*, \mathbf{F}^*$.
Steps:

1: Obtain the execution delay T^l_{ij}, T^s_{ij} and transmission delay T^t_{ij} for all $i = 1, \cdots, N$, $j = 1, \cdots, M$ according to the delay prediction model
2: Model the DNN task partitioning and offloading problem as an MILP problem
3: Solve the MILP problem with standard software packages.

IV. Performance Evaluation

In this section, we present the performance evaluation of our DNN task partitioning and offloading mechanism. For simplification, we regard the processing way determined by our mechanism as **Case 1**. Because the mechanism is among multiple mobile devices and one server, we use four Orange Pi Win Plus as mobile devices and use one computer with CPU i5, 4g RAM, 3.2ghzCPU clock speed as the

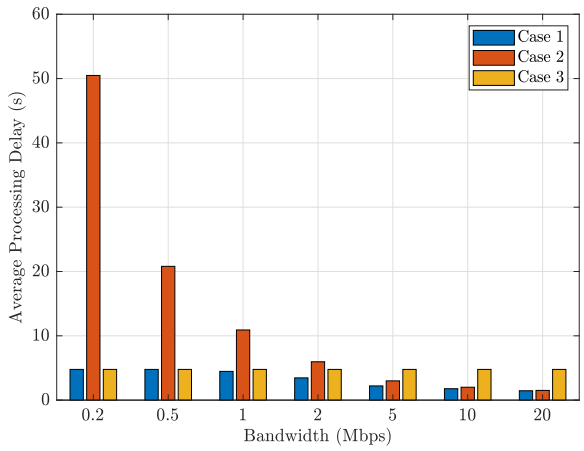

Fig. 1. Average processing delay vs. bandwidth under 3 cases for AlexNet tasks and 0% load level of the server.

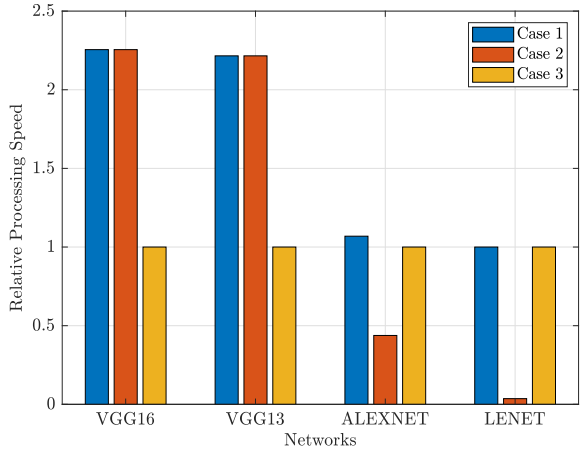

Fig. 2. Relative speed vs. types of DNN tasks under 3 cases for 1Mbps bandwidths and 0% load level of the server.

MEC server. To connect mobile devices with the server, we use Thrift [15] as the communication interface. In Case 1, we obtain our results by real implementation of the delay prediction scheme. After getting the real experimental data about all kinds of delays T^l, T^s, T^t, we use MATLAB to model and solve our MILP optimization problem. In this way, we obtain our numerical results of Case 1.

Before showing our simulation results, we first discuss the following representative task processing mechanisms as baselines for comparison.

- **Case 2:** All tasks are offloaded to the server from mobile devices, and are processed by the server. Most current intelligent applications are processed in this way.
- **Case 3:** All tasks are processed locally at mobile devices.

In Fig. 1, we can find that the average processing delay $\overline{T}$ decreases with the growth of bandwidth under case 1 and 2. And the average processing delay in case 2 decreases more significantly than that in case 1. The reason is that, when the bandwidth is small, the transmission delay is long, which increases the average processing delay. We can also find that the average processing delay remains unchanged under case 3 since the local processing is not affected by the bandwidth. Moreover, the average processing delay in case 1 is always the minimum. It means that, when the bandwidth is small, most tasks are processed locally to reduce the transmission delay. With the increasing of bandwidth, more tasks are partitioned and are offloaded to the server for server's strong computing power.

In Fig. 2, we compare the processing speed when types of DNN tasks are different. We consider 4 different types of DNN tasks, i.e., VGG16, VGG13, ALEXNET, LENET. We assume that, mobile devices have same DNN tasks in one simulation, and the bandwidth is always 1Mbps. We repeat the simulation for 4 times. At each time, the type of DNN tasks is the only difference. Moreover, in each network, we set the processing speed in case 3 as the baseline speed. And we define the related processing speed as the original processing delay divided by the baseline speed. We can see that, Case 1 always has the highest related speed, which demonstrates the superiority of our design under different networks. We can also see that, the related processing speed decreases when the type of DNN tasks changes except for the baseline Case 3. The reason is that, the computing complexities of VGG16, VGG13, ALEXNET and LENET are decreasing. As a result, to process more complex DNN tasks such as VGG16 or VGG13, the execution delay is much longer than transmission delay. Thus, offloading more tasks to the server for server's strong processing power is more time-saving than processing locally. However, to process ALEXNET or LENET whose computing burden is small, execution locally is more time-saving because task offloading will cause relative long transmission delay. Our design trades off the above situations to achieve the maximal processing speed.

In Fig. 3, we investigate the relationship between transmitted data size and bandwidth under 3 cases. In Case 2, all tasks are processed by the server. As a result, the transmitted data is the original data, whose data size is fixed. However, in Case 1, the transmitted data size increases with the growth of bandwidth. In general, when the bandwidth is small, the transmission delay is long, thus our design will prefer processing locally. If the bandwidth becomes larger, mobile devices will prefer to offload more subtasks to the server for strong computing capability. Note that with the growth of bandwidth, the transmitted data size in Case 1 does not increase regularly. The reason is that, task partition patterns between different layers of DNN will make the size of output data different, which leads to different transmitted data size. In Case 3, all tasks are processed locally, thus, there is no transmitted data.

In Fig. 4, we discuss the relationship between average processing delay and load level on the server. We control the

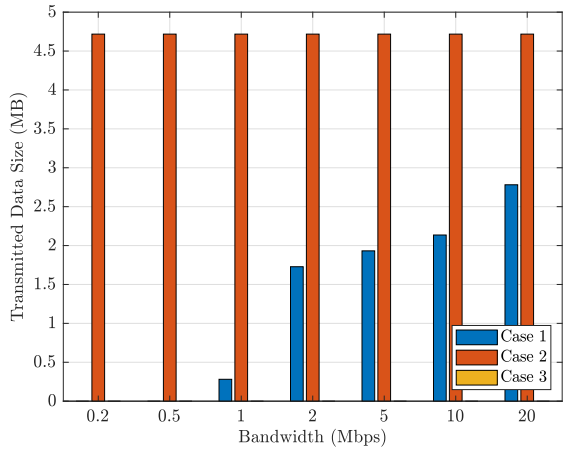

Fig. 3. Transmitted data size vs. bandwidth under 3 cases for AlexNet tasks and 0% load level of the server.

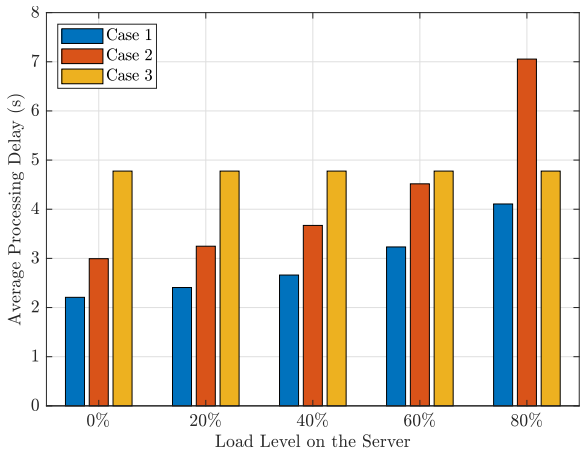

Fig. 4. Average processing delay vs. load level on the server for 5Mbps bandwidths and AlexNet tasks.

server's load level by processing other tasks on the server. We define the server's load level as 0% if there is no other tasks processed by the server. In this figure, we can find that, with the increasing of server's load level, the average processing delay increases in Case 1 and 2, because the increasing of server's load level will decrease the processing speed on the server. However, the average processing delay is unchanged in Case 3 since all tasks are processed locally.

In Fig. 5, we present the relationship between average processing delay and amount of mobile devices under 3 cases. We can find that, with the increasing of mobile devices' amount, the average processing delay increases in Case 1 and 2. This is because there are more offloaded tasks to be processed on the server, which increases the waiting time on the server. However, the average processing delay in Case 3 is unchanged since each mobile device only needs to process its task locally.

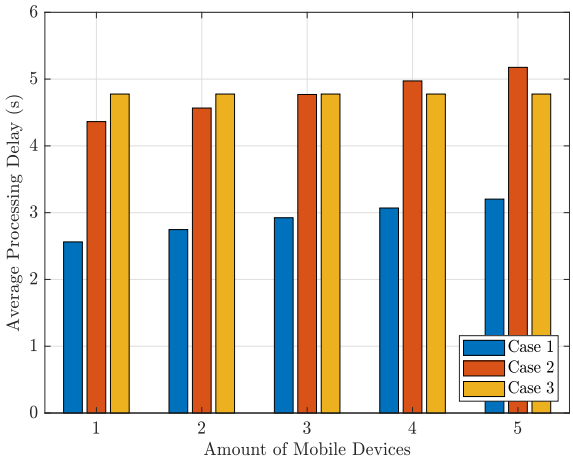

Fig. 5. Average processing delay vs. amount of mobile devices for 2.5Mbps bandwidths, AlexNet tasks and 0% load level of the server.

V. Conclusion

In this paper, we study the DNN task partitioning and offloading problem for MEC. We first proposed a delay prediction model to predict DNN task processing delays under different partition patterns. Based on the delay prediction model, we proposed an MILP-based DNN task partitioning and offloading mechanism to reduce the average processing delay as well as the computing burden of mobile devices. Simulation results show that our mechanism can always achieve the minimal processing delay, compared with traditional processing ways. Evaluations show that our mechanism can achieve up to 90.5% and 69.5% processing delay reduction compared with MEC server only and mobile device only schemes respectively.

Acknowledgments

This work is supported in part by National Natural Science Foundation of China under Grants 61431001, 61872184, 61727802, in part by National Key R&D Program under Grants 2018YFB1004800.

References

[1] C. Lo, Y. Su, C. Lee, and S. Chang, "A dynamic deep neural network design for efficient workload allocation in edge computing," in *2017 IEEE International Conference on Computer Design (ICCD)*, Boston, MA, USA, Nov. 2017, pp. 273–280.

[2] A. Ardakani, C. Condo, M. Ahmadi, and W. J. Gross, "An architecture to accelerate convolution in deep neural networks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 4, pp. 1349 – 1362, Apr. 2018.

[3] Y. Ma, Y. Cao, S. Vrudhula, and J. Seo, "Optimizing the convolution operation to accelerate deep neural networks on fpga," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 7, pp. 1354 – 1367, Jul. 2018.

[4] L. Du, Y. Du, Y. Li, J. Su, Y. Kuan, C. Liu, and M. F. Chang, "A reconfigurable streaming deep convolutional neural network accelerator for internet of things," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 1, pp. 198 – 208, Jan. 2018.

[5] M. Alawad and M. Lin, "Scalable fpga accelerator for deep convolutional neural networks with stochastic streaming," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 4, no. 4, pp. 888 – 899, Oct. 2018.

[6] P. Corcoran and S. K. Datta, "Mobile-edge computing and the internet of things for consumers: Extending cloud computing and services to the edge of the network," *IEEE Consumer Electronics Magazine*, vol. 5, no. 4, pp. 73 – 74, Oct. 2016.
[7] Y. Zhou, L. Tian, L. Liu, and Y. Qi, "Fog computing enabled future mobile communication networks: A convergence of communication and computing," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 20 – 27, May 2019.
[8] L. Liu, Y. Zhou, J. Yuan, W. Zhuang, and Y. Wang, "Economically optimal ms association for multimedia content delivery in cache-enabled heterogeneous cloud radio access networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 7, pp. 1584 – 1593, Jul. 2019.
[9] L. Liu, Y. Zhou, W. Zhuang, J. Yuan, and L. Tian, "Tractable coverage analysis for hexagonal macrocell-based heterogeneous udns with adaptive interference aware comp," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 503 – 517, Jan. 2019.
[10] A. Bozorgchenani, D. Tarchi, and G. E. Corazza, "Centralized and distributed architectures for energy and delay efficient fog network-based edge computing services," *IEEE Transactions on Green Communications and Networking*, vol. 3, no. 1, pp. 250 – 263, Mar. 2019.
[11] C. Yang, Y. Liu, X. Chen, W. Zhong, and S. Xie, "Efficient mobility-aware task offloading for vehicular edge computing networks," *IEEE Access*, vol. 7, pp. 26 652 – 26 664, Feb. 2019.
[12] K. Petersen, A. Kleiner, and O. Stryk, "Fast task-sequence allocation for heterogeneous robot teams with a human in the loop," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, Japan, Nov. 2013, pp. 1648–1655.
[13] M. Xu, S. Wang, J. Tao, and G. Liang, "Research on cooperative task allocation for multiple ucavs based on modified co-evolutionary genetic algorithm," in *2013 International Conference on Computational and Information Sciences*, Shiyang, China, Jun. 2013, pp. 125–128.
[14] F. Shan, J. Luo, W. Wu, and X. Shen, "Delay minimization for data transmission in wireless power transfer system," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 2, pp. 298 – 312, Feb. 2019.
[15] M. A. Messous, H. Sedjelmaci, N. Houari, and S. M. Senouci, "Computation offloading game for an UAV network in mobile edge computing," in *2017 IEEE International Conference on Communications (ICC)*, Paris, France, May 2017, pp. 1–6.