This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LWC.2019.2894703, IEEE Wireless Communications Letters

AAIoT: Accelerating Artificial Intelligence in IoT Systems

Jingyue Zhou, Yihuai Wang, Kaoru Ota, *Member, IEEE,* Mianxiong Dong, *Member, IEEE*

***Abstract*—Existing deep learning systems in the Internet of Things (IoT) environments lack the ability of assigning compute tasks reasonably which leads to resources wasting. In this letter, we propose AAIoT, a method to allocate the inference computation of each network layer to each device in multi-layer IoT system. To our best knowledge, this is the first attempt to solve this problem. We design a dynamic programming algorithm to minimize the response time when weighing the cost of computation and transmission. Simulation results show that our approach makes significant improvements in system response time.**

***Index Terms*—Internet of things; deep learning; resource scheduling.**

I. Introduction

IOT systems capture massive amounts of data from sensors around us. Deep learning is one of the tools we use to extract useful information from these data. The combination of IoT and deep learning brings us many possibilities to explore the real world. There's a lot of work associated with combining these two technologies [1], [2], [3]. Traditional systems that combine IoT with deep learning tend to be cloud-centric. This approach requires a lot of data transferring from local devices to the cloud, which can be time-consuming and has a security risk [4]. With the development of terminal equipment, methods to use the computing resources outside the cloud have been proposed, such as Edge Computing [5] and Fog Computing [6]. Since then, assigning tasks to each device has been a hot topic of research.

Reasonable task allocation improve the efficiency of the system. Xu et al. [7], [8] proposed a resource-efficient edge computing for the emerging intelligent IoT applications. They assigning tasks in an appropriate way to improve the performance. But their approach is not designed for deep learning applications and can not be used in our condition directly. Some studies make good use of edge resources and speed up the inference, such as [9], [10]. But they do not mention how the network is divided. Kang et al. [11] adaptively partitions deep neural networks computation between mobile device and server. They designed a regression model for each layer type to predict the latency and power consumption of the layer running in different devices. They designed an algorithm to cut the networks into two parts base on this information. And they deploy these two parts of networks into mobile device and server respectively to optimize the inference process. On this basis, Li et al. [12] integrated into the early exit technology to further speed up the computation. But both of them only consider the two-layers IoT architecture and cannot accommodate today's increasingly complex system requirements.

In this letter, we propose a neural network segmentation method to optimize the inference process. Different from the existing methods, our method can deploy the neural networks to multi-layer IoT architectures without knowing the details inside the system. We measure the computing and communication ability of each device as the model input. And we design an algorithm to make the optimal strategy using dynamic programming with the principle of minimizing system response time. To the best of our knowledge, this is the first attempt to accurate segmenting neural networks under multi-layer IoT architectures. The simulation results show that our approach is better than other simple computational allocation methods.

II. System Model

In this section, we introduce the model of the system shown in Fig. 1. We allocated the inference computation of the entire network to our devices. The data is collected at the first device. Each device can do computing tasks and deliver the result to the next device. At last, the result calculated by the neural network is communicated back to the first device. The operations of each layer of the network are inseparable, and they depend on the results of the previous layer. We cannot do inference processing in multiple devices at the same time. We use capital letters to represent arrays and use lowercases to represent numbers. We want to allocate n layers deep neural network $L = \{l_0, l_1, ..., l_n\}$ to m IoT devices $D = \{d_1, d_2, ..., d_m\}$. We use $A = \{a_1, a_2, ..., a_n\}$ to represent an allocation strategy, where $a_i \in 1, ...m$ and $a_{i+1} >= a_i$. And a_i means the network layer l_i is assigned to the device d_{a_i}. In our scenario, the data is collected at the first device and the high-layer devices have powerful processing ability. The action of delivering the computation to high-layer devices means sacrificing the transmission time to reduce the calculation time. But deliver the computation to low-layer devices cannot bring benefits. To simplify the model, we specify the layers after l_i can only be assigned to d_{a_i} or IoT devices after d_{a_i}. So there is a limit of $a_{i+1} >= a_i$. Response time and energy consumption

J. Zhou, Y. Wang are with the Embedded System and Internet of Things Technical Lab, School of Computer Science and Technology, Soochow University, China. J. Zhou is also a Visiting Student with the Department of Information and Electronic Engineering, Muroran Institute of Technology, Japan.
E-mail:20164227038@stu.suda.edu.cn, yihuaiw@suda.edu.cn

Kaoru Ota, Mianxiong Dong are with the Department of Information and Electronic Engineering, Muroran Institute of Technology, Japan.
E-mail:{ota, mxdong}@mmm.muroran-it.ac.jp

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LWC.2019.2894703, IEEE Wireless Communications Letters

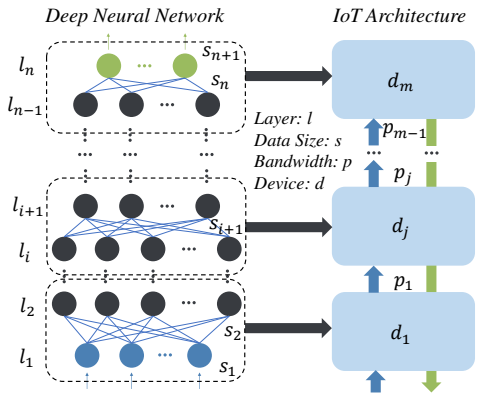


Fig. 1. System model.

are important criteria for IoT systems. We focus on the real-time deep learning application scenarios. In our scenario, all the devices have a steady power supply. Therefore, our model pays more attention to response time. We use the response time $\mathcal{T}$ of the system to judge the allocation strategy. Response time $\mathcal{T}$ has two factors. The first factor is the computing time for each network on the assigned device. The second factor is the time it takes to transfer data between IoT devices. Existing research uses a prediction scheme to predict the computational load of different layers [11], [12]. Our method is designed for the multi-layer IoT systems. There are many factors affect the processing ability and bandwidth of the devices. For example, it is a normal situation when multiple low-level devices share a single high-level device. it will influence the computation time and bandwidth. It is difficult to model these kinds of influence. So we measure the computing and communication ability of each device rather than predict them. This process consumes some resources, but it simplifies our model and makes it more applicable. We denote $C = \{c_1^1, ..., c_i^j, ..., c_n^m\}$ as the layer runtime in each IoT device, where c_i^j is the runtime for neural network layer l_i on IoT device d_j. To calculate the transfer time, we record the size of the output data S for each layer of the network and the bandwidth P between each IoT devices. $S = \{s_1, s_2, ..., s_{n+1}\}$, where $s_1, ..., s_n$ is the input data size of $l_1, ..., l_n$ and s_{n+1} is the output data size of l_n. $P = \{p_1, p_2, ..., p_{m-1}\}$, where p_j is the bandwidth from d_j to d_{j+1}. Above is the definition of all the input and output variables of our model.

Then we will introduce the calculation of the response time $\mathcal{T}$. As mentioned before, $\mathcal{T}$ divide into calculation time $\mathcal{T}_{calcu}$ and transmission time $\mathcal{T}_{trans}$. When we determine the allocation strategy A, the calculation time $\mathcal{T}_{calcu}$ is defined as

$$\mathcal{T}_{calcu} = \sum_{i=1}^{n} c_i^{a_i} \tag{1}$$

It means the sum of time for each layer l_i to compute on the corresponding device d_{a_i}. The transmission time $\mathcal{T}_{trans}$ is defined as

$$\mathcal{T}_{trans} = \sum_{i=1}^{n} \sum_{j=a_{(i-1)}}^{(a_i)-1} (s_i + s_{n+1})/p_j \qquad (a_0 = 1) \tag{2}$$

The first summation is for all the layers. The second summation is for all the data transmission between the two layers. When the adjacent two layers are on different devices, the $a_{(i-1)} <= (a_i) - 1$ is satisfied. This summation will calculate all the transmission consumption between device $d_{a_{i-1}}$ and d_{a_i}. This transmission consumption includes the time cost to transmit the output of l_i from $d_{a_{i-1}}$ to d_{a_i} and the time cost to transmit the network's result from d_{a_i} to $d_{a_{i-1}}$. The s_i in the formula is the size of the l_i's output. s_{n+1} is the size of the network's output, as shown in Fig. 1. And p_j is the bandwidth from d_j to d_{j+1}, where $j \in a_{i-1}, ..., (a_i) - 1$. The $a_0 = 1$ means we do not calculate the transmission consumption before the input of the network in d_1. Finally, the response time $\mathcal{T}$ is define as

$$\mathcal{T} = \mathcal{T}_{calcu} + \mathcal{T}_{trans} \tag{3}$$

In summary, base on the extraction of the essential information, we formulate an optimization problem for layers allocation. That is, when giving each layer's runtime C on each IoT device, the size of the output data S for each layer, and the bandwidth P between each IoT devices, we want to find an allocation strategy A to minimize the response time $\mathcal{T}$.

III. Method

In this section, we introduce the algorithm we proposed to solve the optimization problem formulated in the previous section. After we propose this optimization problem, the intuitive consideration is to use the exhaustion method. We consider the amount of possibility of exhaustion method. This problem is to putting n layers of the neural network into m different devices and allowing empty devices. If we don't allow empty devices, we can simplify this problem as cutting n layers network into m blocks. The devices are in a fixed order. The m blocks of the networks are put in m devices in order. It means to insert $m - 1$ boards in the $n - 1$ gaps between n network layers. When given the layers number n and the devices number m, the amount of possibility is C_{m-1}^{n-1}, where C is the combination number. Then we consider allowing the empty devices. To use the conclusion above, we can assume that each of the m devices originally had one layer. So this problem is equivalent to cutting $n + m$ layers network into m blocks. The number of possible allocation strategies is C_{m-1}^{n+m-1}. The complex calculation of the brute force exhaustive method is $O((n + m - 1)!/(n!(m - 1)!))$. This number will rise dramatically as m and n increase. So we need an approach with less complex.

We propose a dynamic programming algorithm. We denote the state table $H = \{h_1^1, ..., h_i^j, ..., h_{n+1}^m\}$, where h_i^j is the minimum response time when input data of layer l_i has been transmitted into device d_j. With one exception, that is, h_{n+1}^j is the minimum response time when the output of network has been transmitted into device d_j. It is worth noting that when computing h_i^j, we do not specify on which device we compute the l_{i-1} or l_i. We only specify the input data of l_i has been transmitted into d_j. We will explain the reason for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LWC.2019.2894703, IEEE Wireless Communications Letters

this definition later. Then, we consider the calculation of H. When $i = 1$, we calculate the h_1^j by

$$h_1^j = \begin{cases} 0 & j = 1 \\ h_1^{j-1} + (s_1 + s_{n+1})/p_{j-1} & j > 1 \end{cases} \quad (4)$$

When $i = 1$, there is no layer calculation time consumption. The only time consumption is to transfer the input of l_1 from d_1 to d_j and return the result from d_j to d_1. When $j = 1$, no data transfer is required. So the response time $h_1^1 = 0$. When $j > 1$, the response time is the sum of the transmission cost from d_1 to d_{j-1}, that is, h_1^{j-1} and the transmission cost from d_{j-1} to d_j, that is, $(s_1 + s_{n+1})/p_{j-1}$. Then, when $i > 1$, we calculate the h_i^j by

$$h_i^j = \begin{cases} h_{i-1}^1 + c_{i-1}^1 & j = 1 \\ \min(h_{i-1}^j + c_{i-1}^j, h_i^{j-1} + (s_i + s_{n+1})/p_{j-1}) & j > 1 \end{cases} \quad (5)$$

When $j = 1$, we perform all the calculations on d_1, so there is no transmission consumption. The response time is the sum of the runtime from l_1 to l_{i-1}. The h_{i-1}^1 represents the sum of the time consumed to perform the calculations from l_1 to l_{i-2} on d_1. The c_{i-1}^1 is the runtime for calculating l_{i-1} on d_1. After we perform the calculations of l_{i-1} on d_1, the output of l_{i-1} is the input of l_i. It means that there's an input of l_i on d_1. So this calculation is consistent with the definition of H. When $i, j > 1$, there are only two paths to reach the state of h_i^j. The first path is through h_{i-1}^j. It means the input of l_{i-1} is on d_j. It requires c_{i-1}^j to calculate the input of l_i on d_j. The second path is through h_i^{j-1}. It means the input of l_i has been calculated, but this input data is on device d_{j-1}. It requires $(s_i + s_{n+1})/p_{j-1}$ to transmission the data between d_{j-1} and d_j. We choose the path with the shorter response time and calculate the value of h_i^j. After we calculate the H, we can calculate the response time under the optimal strategy by

$$\mathcal{T} = \min_{j \in 1,\ldots,m} h_{n+1}^j \quad (6)$$

It means that we choose the strategy with the shortest response time from all the strategies that finish the whole network. We can calculate the strategy A by Algorithm 1. This algorithm starts at the top layer of the network and gradually infer downward. a_n is the last device used for the optimal strategy. t indicates the current device, so at first $t = a_n$. If $h_{i+1}^t \neq h_i^t + c_i^t$, the calculation for l_i is not on d_t. We will try the lower device until we find the device hosts l_i. We repeat this operation until we find the allocation of all the layers. At this point, we have finished the introduction of our dynamic programming algorithm. H contains $m(n + 1)$ elements, each of which can be calculated in $O(1)$. So the complex calculation of H is $O(mn)$. After we get H, we calculate the strategy A by Algorithm 1. It cost $O(m)$ to find the a_n. In the subsequent steps, it find a strategy from h_i^1 to $h_{n+1}^{a_n}$. This strategy include $n + m$ nodes, and each node can be calculated in $O(1)$. So the complex calculation of Algorithm 1 is $O(m + n)$. The complexity calculation of the overall proposed approach is $O(mn)$.

Algorithm 1: Get Scheme

Input: State Table $H = \{h_1^1, ..., h_i^j, ..., h_{n+1}^m\}$
Output: Allocation Scheme $A = \{a_1, a_2, ...a_n\}$
Compute

$$a_n = \arg \min_{j \in 1,\ldots,m} h_{n+1}^j$$

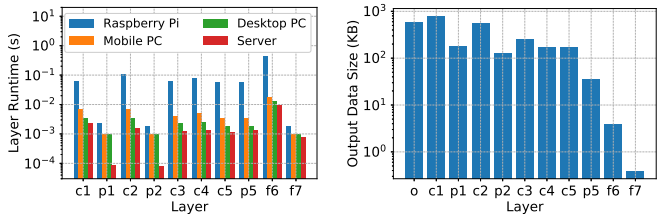
$t = a_n$
for $i \in n - 1, ..., 1$ **do**
  while $h_{i+1}^t \neq h_i^t + c_i^t$ **do**
    $t = t - 1$
  $a_i = t$
return A

Then, we discuss the reasons why our algorithm can find the optimal solution. Our algorithm includes all the possibilities. The most important point is that to get to h_i^j, we have to go through h_i^{j-1} or h_{i-1}^j. The reason is that the computation of the network in our model is continuous and monotonous. And the transmission of data between devices is also continuous and monotonous. It means that there is no case of skipping over a layer or device and can not go backward. So h_i^j is not associated with other states except h_i^{j-1} and h_{i-1}^j. This is why we define h_i^j as the minimum response time when the input data of l_i has been transmitted into d_j. This definition implies that l_{i-1} has been calculated and also indicates that the data has been transferred to d_j. Our experiments also verify that this method gets an optimal solution. The proposed algorithm get the optimal solution. It gets the same solution as the brute force exhaustive method. But the complex calculation is reduce from $O((n + m - 1)!/(n!(m - 1)!))$ to $O(mn)$.

IV. Experiment

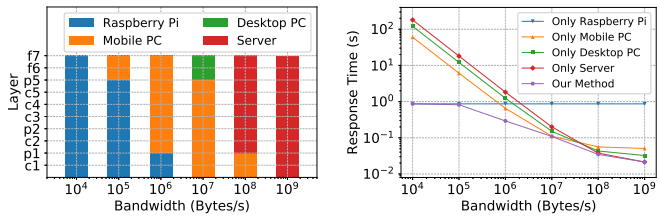
In this section, we present a simulation experiment to demonstrate the performance of our method. Today's neural network structures are mostly hierarchical. Our experiment is designed to approve our method can be used in such hierarchical structures and get a good result. We use our method to optimize the inference process of an image classification task. The input data is RGB image of size $224 \times 224 \times 3$. We design the network structure depend on the AlexNet. The network we designed has 10 layers, that is, $L = \{c1, p1, c2, p2, c3, c4, c5, p5, f6, f7\}$. This network includes 5 layers of convolution $\{c1, c2, c3, c4, c5\}$, 3 layers of pooling $\{p1, p2, p5\}$, and 2 layers of fully connecting $\{f6, f7\}$. Our method speeds up the inference process and does not affect the accuracy. So we do not pay attention to the accuracy in our experiment. There are not many restrictions on the choice of network structure and dataset. Because these choices primarily affect accuracy. We only need the structure is hierarchical, and the amount of computation and data transfer per layer can be measured. The IoT architecture we designed has 4 devices, that is, $D = \{RaspberryPi, MobilePC, DesktopPC, Server\}$. We use Raspberry Pi 3 Model B as $RaspberryPi$. It has a Quad Core 1.2GHz Broadcom BCM2837 64bit CPU and 1GB RAM. The $MobilePC$ has an Intel(R) Core(TM) i5-4210H CPU and 12GB RAM. The $DesktopPC$ has an Intel(R) Core(TM) i7-6700 CPU and 16GB RAM. The $Server$ has an

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LWC.2019.2894703, IEEE Wireless Communications Letters



(a) (b)

Fig. 2. Experimental environment. (a) The layer runtime in each IoT device. (b) The transferred data size between each layer of the network.



(a) (b)

Fig. 3. Simulation result. (a) The allocation strategy with different bandwidths. (b) The response time with different bandwidths.

Intel(R) Core(TM) i7-6700K CPU, a GeForce GTX 1080 GPU and 32GB RAM. As shown in Fig. 2(a), the computing ability of these four devices is rising from the bottom to the top. There is also a variation between devices to calculate different types of layers. For example, the *Server* is particularly adept at calculating $p1$ and $p2$. As shown in Fig. 2(b), the output size of the layer tends to decrease with the inference processing, but there are also some exceptions. The o in Fig. 2(b) represents the original data. At this point, we have collected the necessary data from the real environment.

Then, we verify the validity of our algorithm through simulation. The bandwidths between devices are the same in our simulation, that is $p_i = p_j$, where $i, j \in 1, 2, 3$. We set this network bandwidth as a variable. Our experimental bandwidths cover the current communication mode. The bandwidth of narrow band internet is approximately $10Kbps$. The theoretical speed of 5G achieve $1Gbps$. The simulation result is shown in Fig. 3. We show the allocation strategies for different bandwidths in Figure 3(a). The horizontal axis represents the bandwidth, the vertical axis represents the network layer, and the colors of the bar represent the device we allocate the layer. It is also intuitive that as bandwidth rises, our system tends to allocate computations to higher level devices. In addition, our system tends to make segmentation after $p1$ and $p5$. It related to the fact that the output data size of these two layers is relatively smaller than their neighbor. We show the response time under different bandwidths in Figure 3(b). We draw the response time with a single device as a comparison. We find that the result of our method is the best one. In fact, our algorithm finds an optimal solution under this model. The results of our method are consistent with the exhaustion method. At the same time, Our algorithm saves a lot of computing resources compared with the exhaustive method.

V. Conclusion

This letter provides a quantitative method in computational allocation for deep learning inference tasks in multi-layer IoT systems. Our method allocates the system resources intelligently to reduce response time. In addition, we establish our model under the multi-layer IoT architectures different from existing methods. There are two main contributions in this letter. The first contribution is to establish the model and evaluation criterion base on the computational and transmission time. The second contribution is to design a dynamic programming algorithm to find the optimal solution base on our model. We test our method with a simulation experiment, where we measure the model input from actual devices. Experimental results show that our method provides significant performance improvements in real-world scenarios.

Considering the power consumption will enlarge the application scope of our method. Besides, the condition of multitasking needs intensive research. We will do more related research about these two points in the future.

Acknowledgment

This work is partially supported by JSPS KAKENHI Grant Number JP16K00117 and KDDI Foundation. Mianxiong Dong is the corresponding author.

References

[1] L. Li, K. Ota, and M. Dong, "When weather matters: Iot-based electrical load forecasting for smart grid," *IEEE Communications Magazine*, vol. 55, no. 10, pp. 46–51, 2017.

[2] T. Peng, Y. Wang, T. C. Xu, L. Shi, J. Jiang, and S. Zhu, "Detection of lung contour with closed principal curve and machine learning," *Journal of Digital Imaging*, Feb 2018. [Online]. Available: https://doi.org/10.1007/s10278-018-0058-y

[3] C. Zhang, K. Ota, and M. Dong, "Cooperative positioning optimization in mobile social networks," in *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, Sept 2016, pp. 1–5.

[4] J. A. C. Soto, M. Jentsch, D. Preuveneers, and E. Ilie-Zudor, "Ceml: Mixing and moving complex event processing and machine learning to the edge of the network for iot applications," in *Proceedings of the 6th International Conference on the Internet of Things*. ACM, 2016, pp. 103–110.

[5] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[6] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[7] X. Chen, W. Li, S. Lu, Z. Zhou, and X. Fu, "Efficient resource allocation for on-demand mobile-edge cloud computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 9, pp. 8769–8780, Sept 2018.

[8] X. Chen, Q. Shi, L. Yang, and J. Xu, "Thriftyedge: Resource-efficient edge computing for intelligent iot applications," *IEEE Network*, vol. 32, no. 1, pp. 61–65, Jan 2018.

[9] H. Li, K. Ota, and M. Dong, "Learning iot in edge: deep learning for the internet of things with edge computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, 2018.

[10] S. Teerapittayanon, B. McDanel, and H. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 328–339.

[11] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGPLAN Notices*, vol. 52, no. 4, pp. 615–629, 2017.

[12] E. Li, Z. Zhou, and X. Chen, "Edge intelligence: On-demand deep learning model co-inference with device-edge synergy," *arXiv preprint arXiv:1806.07840*, 2018.