Distributed Processing Method for Deep Learning in Wireless Sensor Networks

Karin Umeda¹, Takashi Nishitsuji², Takuya Asaka², and Takumi Miyoshi³

¹ Graduate School of Systems Design, Tokyo Metropolitan University  
6-6 Asahigaoka, Hino-shi, Tokyo 191-0065, Japan  
² Faculty of Systems Design, Tokyo Metropolitan University  
6-6 Asahigaoka, Hino-shi, Tokyo 191-0065, Japan  
³ College of Systems Engineering and Science, Shibaura Institute of Technology  
307 Fukasaku, Minuma-ku, Saitama-shi, Saitama 337-8570, Japan

Abstract: In wireless sensor networks, distributed processing technology for deep learning that utilizes edge computing and mobile terminals has been attracting interest; however, with the expansion of the use of wireless sensor networks, the amount of data and the increase in server load are issues, especially for deep learning. Although the existing studies which improve the data processing speed and latency reduction of distributed processing, there is still a problem in which the amount of communication and the load on the server increase. In this paper, we propose load balancing algorithm for deep learning to reduce the communication volume and server load.

Keywords: Wireless Sensor Network, Deep Learning, Distributed Processing

Classification: Network System

References

[1] E. Li, Z. Zhou, and X. Chen, “Edge intelligence: On-demand deep learning model co-inference with device-edge synergy,” MECOM ’18, pp. 31–36, Aug. 2018.

[2] J. Mao, Z. Yang, W. Wen, C. Wu, L. Song, K. W. Nixon, X. Chen, H. Li, and Y. Chen, “MeDNN: A distributed mobile system with enhanced partition and deployment for large-scale DNNs,” IEEE/ACM IC-CAD 2017, pp. 751–756, Nov. 2017.

[3] J. Motoyama, R. Ooka, T. Miyoshi, T. Yamazaki, and T. Asaka, “Distributed processing allocation of machine learning in wireless sensor networks,” ICCE-TW 2020, no.A2–2, Sep. 2020.

[4] L. N. Huynh, Y. Lee, and R. K. Balan, “DeepMon: Mobile GPU-based deep learning framework for continuous vision applications,” 15th ACM Int. Conf. Mobile Syst., Appl. Services (MobiSys 2017), pp. 82–95, June 2017.

[5] K. Umeda, T. Nishitsuji, T. Asaka, and T. Miyoshi, “Distributed processing method for deep learning on wireless sensor networks,” International Conf. on Emerging Technologies for Commun. (ICETC 2020), no.N1–3, Dec. 2020.
1 Introduction
As the use of wireless sensor networks expands, the amount of data acquired from sensor will also increase, so the amount of communication is expected to increase. In addition, since the acquired data is processed by the server, the increasing server load also becomes a significant issue for large scale data processing such as deep learning.

Therefore, a distributed processing method for deep learning using edge computing and mobile terminals is currently being proposed [1, 2]. A distributed processing method for deep learning using mobile terminals and edge servers, and a parallel processing method for convolutional layers using multiple mobile terminals have been proposed. Furthermore, a method using the GPUs (Graphic Processing Unit) of a mobile terminal has also been proposed [4]. However, these studies focus on latency and processing speed, and do not take traffic and server load into consideration.

Therefore, we propose a distributed processing method for deep learning that considers the processing capacity of the sensor and the communication environment for the purpose of reducing the server load and communication volume.

2 Related Study
Li et al. proposed a method to determine the size of DNN (Deep Neural Network) according to the processing and to divide the calculation process of DNNs between mobile terminal and edge server [1]. When the execution time of each layer of DNN and the data size of the output result were examined, it was found that there was no proportional relationship. In other words, it is possible to offload a layer with a large amount of calculation to the server with low overhead, and it is possible to reduce the end-to-end delay. The layer with a large amount of calculation is used as the division point between the edge server and the mobile device calculations, the calculations before this division point are offloaded to the server, and the remaining calculations are performed on the mobile terminal. In addition, delays and computational complexity are reduced by choosing the size of DNN for the application.

It is possible to reduce the data transfer delay by appropriately dividing DNN operation based on the available bandwidth between the edge server and the mobile terminal. However, depending on the bandwidth and delay, all DNN may be executed on the edge server, and the load on the edge server may not be reduced.

Mao et al. proposed a technique for parallel processing of convolutional layers using multiple mobile terminals [2]. One of the mobile terminals becomes the Group Owner (GO), and the convolutional layer calculations are processed according to the assignments determined by the GO. When each terminal finishes the computation process, the result is transferred to the GO, which then uses it to update the parameters and assign the computation process of the next convolution layer. By repeating this, it is possible to process intermediate layer operations quickly.
Since communication is performed between mobile terminals every time the convolutional layer is calculated, the amount of the overall network communication increases. In addition, because GO terminals frequently communicate and play the role of assigning computations, their power consumption is increased.

Motoyama et al. proposed a method to appropriately distribute the machine learning calculation processing on the route to the sink node in the wireless sensor network and equalize the total calculation processing of each node [3]. First, the order of allocation of calculation costs is determined by the descending order of the number of hops to the sink node. Calculation costs are allocated equally to each node based on the previously determined order. Nodes with overlapping paths are determined by dividing by the number of nodes with overlapping paths with multiple machine learnings.

With a conventional method, the calculation results are sequentially transmitted to the server, and the communication associated with the calculation-processing is concentrated on the server and some mobile terminals, so the amount of communication increases and the load on the server does not change.

Therefore, we propose a method of distributing and allocating the computational processing, taking into account the processing capacity of the sensor and the communication environment.

The proposed method uses only sensor nodes for distributed processing, while the conventional methods [1, 2] use different devices such as mobile nodes and edge nodes. In reference [3], the amount of processing is treated as a continuous quantity, which is different from the proposed method.

3 Distributed Processing Method

3.1 Overview

In this paper, we propose a method to divide the arithmetic processing of the intermediate layer of the deep learning according to the processing capacity of the sensors and the communication environment in order to reduce the amount of communication and the load on the server. The network structure and data transfer path are predetermined, and by clarifying the amount of computation for each layer in advance, the DNN calculation is divided so that the variance of the amount of calculation allocated to each node is minimized. Figure 1 shows a schematic diagram of the proposed method.

First, the server side determines the processing capacity of the sensor and the communication environment in advance. Next, based on that information, the intermediate layer of DNN is divided into multiple parts and assigned to the sensors. For example, in Fig.1, the first node $n_1$ obtains input data and process two convolutional layers and one pooling layer. The calculation result of $n_1$ is input to the second node $n_2$, and $n_2$ process two convolutional layers. By repeating this, the whole process of DNN is completed. Here, we assume the data is transferred with multi-hop communication between nodes. Since the calculation process of deep learning is divided and executed in order in the sensor network, the proposed method can reduce the...
Assign the layers in advance according to processing capacity.

This allocation will be reviewed on a regular interval, such as once a week.

This proposed method is suit for applying real-time processing, e.g., a crime prevention / monitoring system such as intruder prevention.

3.2 Formulation
We formulated the proposed method as minimizing the variance of the computational complexity allocated to each node. Here, we assumed the following constrains: data is acquired only at start node, the processing capacity of each node is the same, no other processing is performed, and the transfer route is fixed.

Define the load of \( j \)-th node as:

\[
L_j = \sum_{i=1}^{R} \sum_{k=1}^{N_i} r_i x_{ikj} l_{ik},
\]

where \( R \) is the number of task, \( r_i \) is the number of processing request of \( i \)-th task per unit time, \( N_i \) is the number of layers in the \( i \)-th task, \( l_{ik} \) is the computational load of \( k \)-th layers of \( i \)-th task, and \( x_{ikj} \) is defined as,

\[
x_{ikj} = \begin{cases} 
1 & \text{(when } k \text{-th layer of } i \text{-th task is allocated on } j \text{-th node)} \\
0 & \text{(otherwise)} 
\end{cases}
\]

Then, we solve the optimization problem as follows:

\[
\min \quad v(L_j) \\
\text{s.t.} \quad x_{ikj} \in \{0, 1\} \\
\quad \sum_{j=1}^{M} x_{ikj} = 1 (k = 1, \ldots, N_i, \forall i, j) \\
\quad \sum_{a_1=1}^{j} x_{ika_1} \geq \sum_{a_2=1}^{j} x_{ika_2} (\forall i, j, a_1 < a_2).
\]

Here, \( v(L_j) \) is the variance of the node load. According to Eq.(3), it indicates “assigns each layer to one node somewhere”, and according to Eq.(4),
it indicates “there is no layer overtaking when allocating nodes”. For example, the first and third layers are not assigned to one node, but the first and second layers are assigned in order.

In this paper, we approximated the number of multiplications to the amount of calculation of layers of DNN because it occupies most of the DNN calculations. Let \( l_{ik} \) be the number of multiplications in the \( k \)-th layer of \( i \)-th task, and define it as follows:

\[
l_{ik} = F_{ik}^2 K_{ik} O_{ik} (W_{ik}/s_{ik})^2,
\]

where \( F_{ik} \) is the size of the filter, \( K_{ik} \) is the number of color channels of the image, \( O_{ik} \) is the number of calculation result data, \( s_{ik} \) is the width to shift the filter, \( W_{ik} \) is assumed that the size of the input data.

### 3.3 Processing Allocation Algorithm
The optimization problem in the previous section has the problem that the computational time becomes long when there are many task and nodes. Therefore, we propose an algorithm with the two preconditions of “the number of nodes is fixed” and “the nodes perform only calculation processing”.

**STEP1:** Calculate the total number of multiplications \( C \) per unit time by the following.

\[
C = \sum_{i=1}^{R} \sum_{k=1}^{N_i} r_i l_{ik}.
\]

**STEP2:** Calculate the average number of multiplications \( C_a \).

\[
C_a = C/M.
\]

**STEP3:** Calculate the number of multiplications \( C_j \) assigned to node \( j \). Allocation is given priority to the layer with the highest number of multiplications. When allocating more times than \( C_a \), select the layer with the least number of multiplications from the layers that may be assigned next.

\[
C_j = \sum_{i=1}^{R} \sum_{k=p}^{q} l_{ik} (0 \leq p, q \leq N_i).
\]

Since this algorithm preferentially allocates layers with a large number of multiplications, it is easy to adjust the layer allocation, and it is possible to allocate a number close to the average number of multiplications. However, since there is no clear standard as to how many multiplications assigned to a node can exceed the average number of multiplications, the variance may increase depending on the allocation method.

### 4 Case Study
The above algorithm is used to determine the allocation to each node. This time, we used DNN of three object detections, AlexNet, VGG16, and GoogLeNet. In this paper, the request per unit time of Alexnet is \( r_1 \), that of VGG16 is \( r_2 \), and that of GoogLeNet is \( r_3 \). As an example of allocation, consider the
case where \( r_1 = 2, \ r_2 = 1, \ r_3 = 1 \) and the number of nodes is 5, and the case where \( r_1 = 1, \ r_2 = 2, \ r_3 = 3 \) and the number of nodes is 7. The results of the allocation are shown in Fig. 2 and 3.

Since there are many layers with a small number of multiplications, the number of multiplications can be assigned close to \( C_a \). In deep learning, there are layers with a large number of multiplications in the first half, and many layers with a small number of multiplications in the second half. That is, the last node is assigned a large number of layers with a small number of multiplications, which is lower than the average number of multiplications. On the contrary, since the layer with a large number of multiplications is assigned to the nodes in the first half, many nodes exceed \( C_a \).

5 Conclusion

In this research, we proposed a method to distribute the calculation processing of deep learning to the sensors in order to reduce the load on the server. In the future, it is necessary to consider allocations that take into account the communication environment, sensor processing, and dynamic situations.