室蘭工業大学
学術資源アーカイブ
Muroran Institute of Technology Academic Resources Archive

Sustainable CNN for Robotic: An Offloading Game in the 3D Vision Computation,

メタデータ	言語: eng 出版者: IEEE 公開日: 2020-11-16 キーワード (Ja): キーワード (En): Sustainable computing, convolutional neural network (CNN), 3D scene understanding, robotic 作成者: LI, Liangzhi, 太田, 香, 董, 冕雄 メールアドレス: 所属:
URL	http://hdl.handle.net/10258/00010309

Sustainable CNN for Robotic: An Offloading Game in the 3D Vision Computation

Liangzhi Li, *Student Member, IEEE,* Kaoru Ota, *Member, IEEE,* Mianxiong Dong, *Member, IEEE*

Abstract—Three-dimensional (3D) scene understanding is of great significance to many robotic applications. With the huge development of the deep learning methods, especially the convolutional neural network (CNN), 3D robotic vision has achieved a satisfactory performance. However, in most scenarios, sustainability becomes a severe problem, and few existing approaches pay enough attention to energy consumption. In this paper, we propose an energy-aware system for sustainable robotic 3D vision. Our contributions mainly include: 1) an effective CNN model for the 3D scene understanding; 2) an offloading strategy to make the deep model more sustainable. First, we design a deep CNN model to analyze the 3D point cloud data. The proposed model contains 92 layers for a state-of-the-art recognition accuracy, which, however, bring a big burden to the computing hardware. Then, we formulate this deep learning computation problem as a non-cooperative game, and adopt a heuristic algorithm to balance the local computing and cloud offloading, in order to obtain an optimal solution, in which both the efficiency and energy-saving are taken into account. Simulations demonstrate that our approach is robust and efficient, and outperforms the state-of-the-art in several related tasks.

Index Terms—Sustainable computing, convolutional neural network (CNN), 3D scene understanding, robotic.

1 Introduction

Computer vision is always a hot area in the past few decades. It is an essential component in many autonomous systems, such as the robotic, self-driving, indoor localization, action recognition, etc. During its long history, a lot of scene understanding methods have been proposed to analyze the input images for their semantic meanings, e.g., K-means, decision tree, Support Vector Machine (SVM) and convolutional neural network (CNN). Among all these approaches, deep learning methods, including CNN, are in evidence due to their strong abilities of automatic feature extraction and pattern recognition [1]. CNN and other deep learning methods are widely used in various fields. As the most successful application, these deep models have been the de facto standard for computer vision applications, as they have shown a huge advantage on the recognition accuracy.

Meanwhile, with the recently emerged three-dimensional (3D) sensing technologies [2], computer vision has evolved to the next stage. Compared to the traditional two-dimensional (2D) sensors, 3D sensors are able to obtain the spatial information as well as the RGB data, and they are more robust and stable [3]. Therefore, 3D image is inherently a better choice for these autonomous systems. However, 3D image processing also brings a huge burden to the underlying hardware when using the state-of-the-art deep learning models, which require a lot of computation. This has become a severe problem to the devices which have limited battery. We have proposed a distributed robotic system in [4]. This system can perform efficient object detection and recognition on 3D input images. However, the proposed system leads to serious energy problems in the wirelessly-connected robots, even with a simple CNN model. As the recently proposed models grow deeper and deeper [5], [6], it is the trend to make more complicated models for better recognition precision. Energy-saving has become indispensable for the robotic applications. We find it necessary to completely modify our model in two aspects. One is to make a larger model which contains dozens of neural layers for better accuracy; the other one is to offload the computation from the energy-limited robots to the cloud servers, in order to make a sustainable robotic system.

For the former one, we propose a new CNN model in this paper. This model is based on our former work [4]. We extend the existing eight-layer model to 92 layers, in order to achieve higher precision. To adapt the CNN model into the offloading game, we deploy the same models on all the mobile robots and central servers, and the servers can restore the calculation from any breakpoints with the intermediate values uploaded by the mobile robots.

Computation offloading is important to build an energy-efficient system. This is because that the battery-limited devices always care about their energy consumption and prefer to save the battery for longer usage time. And on the other hand, the computation burden does exist, and has to be processed, therefore, can only be offloaded to some other devices. In this work, we focus on a multi-user offloading strategy, which is designed to simulate an actual robotic application, where multiple robots exist and attempt to analyze the captured 3D images, as shown in Fig. 1. The adopted robots in our system are equipped with depth sensors, and can obtain the 3D information of its current field of view (FOV). For energy-saving, the offloading controller will decide the computation allocation among the mobile robots and the cloud servers. These robots need to know to what degree they should compute the deep model on their own and leave the remaining part to the cloud servers.

• *Authors are with the Department of Information and Electronic Engineering, Muroran Institute of Technology, Japan. E-mail:{16096502, ota, mxdong}@mmm.muroran-it.ac.jp*

Manuscript received xx xx, 2017; revised xx xx, 2017.

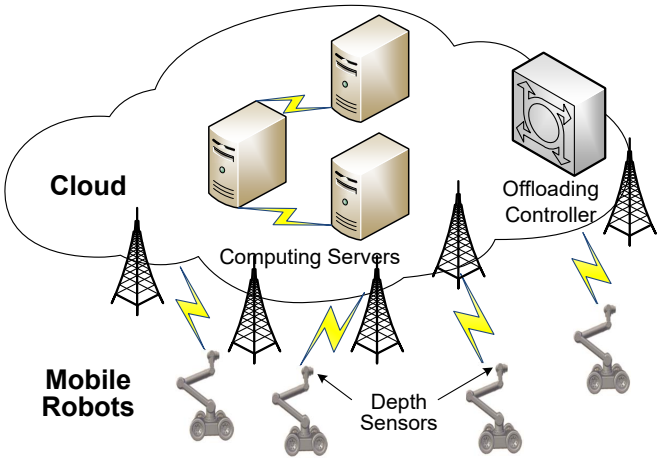


Fig. 1. Computation offloading for the robotic 3D CNN models. Each robot is equipped with depth sensors, and can capture the 3D data of its current field of view (FOV). For energy-saving, the offloading controller will decide the computation allocation among the mobile robots and the cloud servers. Robots only calculate for part of the CNN model, and upload the intermediate result to the cloud by wireless communications.

Each robot, or "player" in other words, can select among the following three choices, i.e., directly uploading the raw data, calculating the whole model or part of it. The remaining part of the CNN model, which is not processed by the mobile robots, will be calculated by the central servers.

The main contributions of our work include:

- We design a complex CNN model for the 3D scene understanding task. Results demonstrate it has a higher recognition performance.
- We formulate the deep learning computation problem as a non-cooperative game, and adopt a heuristic algorithm to balance the local computing and cloud offloading, for better energy-saving and sustainability. We propose two different models to respectively represent the computation and communication cost.
- We conduct extensive experiments to test the performance, on both the scene understanding accuracy and the system cost-saving effect.

2 Related Works

As the paper mainly consists of two subtopics, i.e. 3D scene understanding and computation offloading, some related works in these areas will be introduced in this section.

2.1 3D Scene Understanding

The rise of 3D sensing technologies gives a huge boost to computer vision field. Using 3D sensors, robots can obtain much more information than the traditional cameras, which can only output simple RGB data. However, 3D images also bring a big problem, i.e., the well-developed scene understanding methods, which are designed for 2D images, cannot work well in the new 3D world. Many researchers are working on this topic.

One of the earliest attempts on applying CNN in 3D image recognition is the CRNN [7], which is a combination of the recursive neural network (RNN) and CNN. The CNN part is responsible for the extraction of lower-level features, and RNN is used to organize the lower layers into one hierarchical operation. Su et al. try another approach by rendering the 3D shapes into a set of multi-view images [8]. The authors present a new CNN architecture to utilize the data from multiple views. Eitel et al. design their model as two separate streams, one for the RGB channel and the other one for the depth channel [9]. This network can fuse RGB and depth data, and output the final recognition result based on the combination of these two streams. Maturana and Scherer study the specific effects of several different designs of their CNN architecture [10]. The input data of their CNN model is the point cloud from LiDAR sensors. One distinctive point of this work is that the model conducts the feature extraction and recognition directly on the raw 3D data, rather than converting it into 2D images. We also work out a view-invariant CNN model with several specially-designed multi-task loss functions [4]. This model can extract similar features of a 3D object, even from different views. This is an essential ability when deploying robots in the real world, which is highly disordered and changeable.

In this paper, we continue to improve our scene understanding model, which can achieve a higher accuracy, but brings a significant increase of the model complexity and computational cost. To solve this problem, we work out a mathematical model of its computation cost, and successfully adapt the proposed CNN to an offloading game for better sustainability.

2.2 Computation Offloading and Sustainable Learning

Computation offloading is a novel paradigm to balance the calculation burden among the mobile devices and central servers, and to decrease the energy cost of the battery limited devices, especially in the era when a large size of data are captured by sensors, mobile devices, robots, etc. [11], [12], [13]. Many encouraging works have been proposed in this area.

Kosta et al. [14] build a framework called ThinkAir for developers to migrate the mobile applications to the central cloud servers. It is one of the first attempts of mobile offloading, and shows a good effect on energy-saving. Wen et al. [15] obtain the energy-optimal policy by solving two constrained optimization problems. One is how to set the CPU clock frequency, and the other one is how to schedule the network communication. The optimization objective is to minimize the energy cost within time delay. Munoz et al. [16] work on the tradeoff between energy consumption and latency, and provide a joint optimization model. They also design a method to calculate the optimal strategy to allocate the computational load among the mobile devices and the cloud. Kwak et al. [17] jointly consider three dynamic problems in the actual application scenarios, and present an algorithm using Lyapunov optimization to minimize the energy cost of CPU and network communication while meet given delay constraints. The research of Yang et al. [18] is one of the first work allowing the dynamic partitioning and computation instance sharing among multiple users in the cloud. And their framework is highly scalable and the results demonstrate its performance. Chen et al. [19] focus on a multi-user offloading problem. The authors study a mobile-edge scenario where multi-users and multi-wireless

channels exist. They prove that this game always has a Nash equilibrium, and propose an algorithm to achieve it. Dynamic cloudlets based methods are used to decrease the energy cost of the wireless communications brought by the edge computing [20], and the results demonstrate their efficiency. The works on remote cloud servers or heterogeneous core processors [21], [22] also provide some solutions to address the energy problem in computation offloading.

However, to our best knowledge, there is few offloading approaches focused on the energy-saving problem of the deep learning models. Although the calculation cost of the deep models is very high, the current mainstream is still to merely find some ways to decrease this cost, using some methods like decreasing the number of weights, simplifying the deep models, deleting some layers, etc., rather than offloading the calculation tasks to the cloud. Guo et al. [23] give a model surgery method to compress the CNNs, which can significantly reduce the network complexity. There are also other similar approaches, such as [24], [25], [26]. In addition, although there exist several offloading methods designed for deep models, they are not focused on the energy problem. For example, Zhang et al. [27] design a privacy-preserving offloading model for the security problem; Das et al. [28] propose a distributed Stochastic Gradient Descent (SGD) algorithm to accelerate the training speed; Ran et al. [29] make a prototype of mobile deep learning application using offloading, in order to implement a real-time Augmented Reality (AR) mobile application.

Therefore, we find it necessary to find another way to realize the sustainable deep learning for 3D scene understanding tasks. In this paper, we successfully formulate the offloading game and design an algorithm for computation offloading.

TABLE 1
Main Notations

Notation	Description
$\mathcal{P}$	Set of the game players $\mathcal{P} = \{p_1, ..., p_r\}$
$\mathcal{D}$	Set of the player decisions $\mathcal{D} = \{d_1, ..., d_r\}$
$\mathcal{G}$	Set of the possible decisions $\mathcal{G} = \{0, 1, 2, ..., g_{\max}\}$
r	The total number of the players.
b	Bandwidth of the wireless network.
γ_a	Uploading rate of the arbitrary player a.
ρ	Transmission power.
g_i^c	Channel gain between player a and base station c.
s	Uploading data size.
C_a^{comm}	Total communication cost caused by player a
C_a^{comp}	Total computation cost caused by player a
C_a^{local}	Local-computing factor
C_a	Overall cost caused by player a
t_a^{comm}	time cost of the network transferring.
e_a^{comm}	energy cost of the network transferring.
l	Layer number of the adopted deep model.
$\mathcal{M}$	Set of the input image numbers. $\mathcal{M} = \{m_1, ..., m_r\}$
$\mathcal{E}^{\text{layer}}$	Set of the layer-wise energy cost. $\mathcal{E}^{\text{layer}} = \{e_1^{\text{layer}}, ..., e_l^{\text{layer}}\}$
$\mathcal{T}_{\text{player}}^{\text{layer}}$	Set of the layer-wise time cost for the players. $\mathcal{T}_{\text{player}}^{\text{layer}} = \{t_{player,1}^{\text{layer}}, ..., t_{player,l}^{\text{layer}}\}$
$\mathcal{T}_{\text{server}}^{\text{layer}}$	Set of the layer-wise time cost for the servers. $\mathcal{T}_{\text{server}}^{\text{layer}} = \{t_{server,1}^{\text{layer}}, ..., t_{server,l}^{\text{layer}}\}$
$\mathcal{S}^{\text{layer}}$	Set of the layer-wise output data size. $\mathcal{S}^{\text{layer}} = \{s_0^{\text{layer}}, s_1^{\text{layer}}, ..., s_{l-1}^{\text{layer}}, 0\}$

3 Overview

3.1 System Framework

The framework of the proposed system is presented in Fig. 2. Our system mainly consists of three components, including the mobile robots, the offloading controller, and the computing servers. As shown in Fig. 1, the controller and servers are located in the remote cloud, and the mobile robots are deployed in different spots to conduct various tasks, such as exploring, rescuing, transport, etc. We choose the cellular network to connect the mobile robots with the remote cloud servers, because this kind of wireless network is commonly used and can save the extra expenditure regarding the network infrastructure and deployment. Robots use their onboard depth sensors to capture the 3D images from their FOVs, and save the data in their memory for the following recognition process. The controller is responsible for the control of the whole offloading process. It not only cares about the load allocation, but also the communication cost. Although both mobile robots and computing servers have the ability to perform the forward computing of the CNN models, the servers are more powerful in computing ability than the energy-limited robots, and also has larger storage including memory, hard disk, and database, which can keep the records for further analysis.

In each scene understanding scenario, the robots will wait until that offloading controller has made a decision, and receive this decision using their interfaces of wireless networks. According to the decision, the robots will either pull the images from their memory and use their CNN modules to calculate part of the installed CNN model, or simply upload the unprocessed images. The intermediate values and the raw data are transferred to the cloud servers for further processing, and the servers will complete the remaining computing and output the recognition results.

We will introduce the CNN module deployed in the robots and computing servers in the next section. The computation offloading algorithm, which is the focus of this paper, is detailed in Section 4. In this paper, we use calligraphic-font letters to represent sets, e.g. $\mathcal{P}$, and notations are listed in Table 1.

3.2 Scene Understanding Model

We have proposed a 3D scene understanding framework in [4], which includes two sub-networks respectively for object detection and recognition. As we focus on computation offloading in this paper, we simplify this model for better understanding, as shown in Fig. 3a. This CNN model consists of six layers, including four convolutional layers and two full-connected layers. In our past work, we have shown that this model has a good performance in recognition speed and accuracy. However, in actual applications, we find a severe energy problem due to intense computing needs, which can quickly use up the battery.

In order to further improve the recognition performance, and more importantly, address the problem of energy consumption, we propose a newly-designed CNN model in this section. As shown in Fig. 3b, the modified model is much deeper than the former one, which brings more potential to this model for even higher performance. This model has

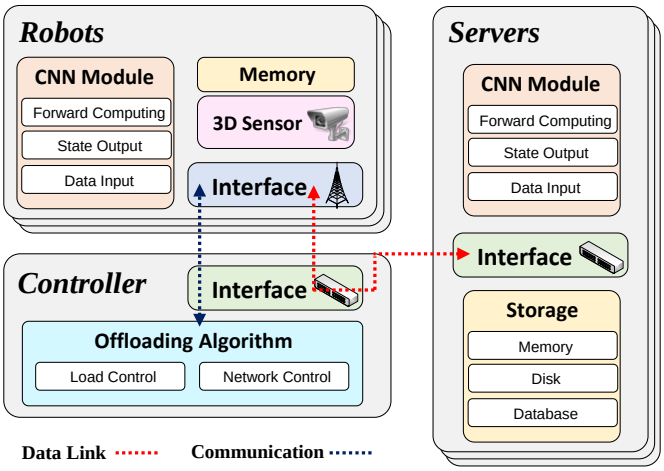


Fig. 2. System framework. The proposed system mainly consists of three components, i.e., the mobile robots, offloading controller, and the computing servers.

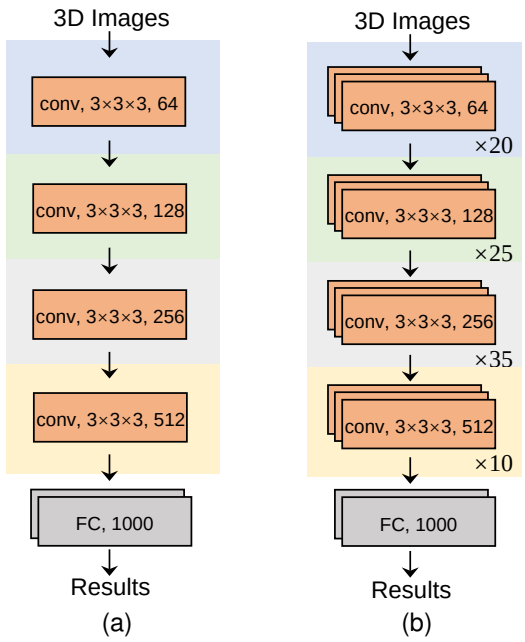


Fig. 3. The comparison between the newly-designed CNN model and the one of our past work. (a)The existing model, which contains six layers. (b)The modified model, which contains 92 layers.

90 convolutional layers and two fully-connected layers. We divide them into four virtual classes with the layer number as 20, 25, 35, 10, and each of them corresponds to one convolutional layer in the former CNN model.

To adapt this model into the offloading game, we need to clarify the computation cost of each layer, including the energy consumption and time cost, and import these information into the following offloading algorithm. Because over 99% operations in convolutional layers are multiply-and-accumulate (MAC) operations [30], [31], they account for most of the energy consumption and time cost nearly in all of the state-of-the-art CNN models. Chen et al. have implemented an evaluation framework in [32] to estimate the energy consumption of each layer in the deep models. And in [33], Chen et al. give a detailed performance breakdown of the AlexNet model [34]. In their paper, they give out many useful data regarding the layer-wise consumption, including both energy cost and time cost. Although their work is focused on the traditional 2D CNN models, but the relationship between time cost and energy consumption still illuminate the way of our offloading strategy design, because both of them are caused by the same factor. Therefore, we can infer the rough relationship between time cost and energy consumption in our newly-designed 3D CNN model, which is very useful in designing our cost estimation models. Based on the aforementioned theories, we can now suppose an estimation of the layer-wise energy cost $\mathcal{E}^{\text{layer}} = \{e_1^{\text{layer}}, ..., e_l^{\text{layer}}\}$ of the adopted model. Then we can give a rough estimation of two different time cost on robotic hardware and computing servers, i.e., $\mathcal{T}_{\text{player}}^{\text{layer}} = \{t_{p,1}^{\text{layer}}, ..., t_{p,l}^{\text{layer}}\}$ and $\mathcal{T}_{\text{server}}^{\text{layer}} = \{t_{s,1}^{\text{layer}}, ..., t_{s,l}^{\text{layer}}\}$, respectively for the player and server. In addition, the data size $\mathcal{S}^{\text{layer}} = \{s_0^{\text{layer}}, s_1^{\text{layer}}, ..., s_{l-1}^{\text{layer}}\}$ can be directly obtained by the model structure. All of these information, including energy cost, time cost, and data size, are adopted in the following cost estimation model, which will be detailed in the next section.

4 Offloading Strategy

In this section, we will give several cost models, and based on that, formulate the proposed computation offloading strategy.

We denote the robots, which are also the players in the offloading game, as $\mathcal{P} = \{p_1, ..., p_r\}$, where r is the total number of the players, and represent the player decisions as a set $\mathcal{D} = \{d_1, ..., d_r\}$, where d_a is the final decision of player a.

As the robots may belong to different individuals and organizations, we suppose each player wants to find the most energy-efficient and time-saving solution for itself, in order to make its onboard computing system more sustainable. But their own optimal solutions may be contradictory, and cannot exist at the same time. Therefore, it is necessary to set a controller to look for the global optimal solution, which can minimum the overall cost of the whole system. This controller should respond the requests from the players, and we must design an algorithm for it to decide which requests should be satisfied, and which should be not. We will demonstrate the proposed heuristic algorithm can achieve good performance in the experiment section.

In order to design the computation offloading algorithm, we need to give out the estimation models regarding the communication cost and the computation cost. The costs mainly contain two aspects, i.e., the energy consumption, which is vital for the battery-limited devices, and the time cost, which brings the delay of the response and deteriorate the overall performance.

4.1 Communication Cost Estimation

First, we need to find a general model for the wireless transfer rate, because all the robots communicate with the remote cloud via the wireless networks, such as the cellular

networks. Then the uploading rate of one arbitrary robot p can be expressed as

$$\gamma_a = b \log_2 \left(1 + \frac{\rho_a g_a^c}{\rho_b + \sum_{i \in \mathcal{P}, d_i < g_{\max}} \rho_i g_i^c} \right) \tag{1}$$

where γ_a represents the transfer rate of robot a, b is the bandwidth of the wireless network, ρ_a is the transmission power of player a, and g_a^c represents the channel gain between player a and base station c. According to the wireless interference model, the channel gain g_a^c is related to the distance between player a and its corresponding base station c, and can be expressed as

$$g_a^c = \frac{1}{{L_{a,c}}^{\alpha}} \tag{2}$$

where L is the distance between player and base station, and α is the path loss factor. Since we focus on an offloading game for deep learning applications, we can omit several communication details to simplify the equations. We suppose each player a has the same transmission power. Therefore, Eq. (1) can be presented as

$$\gamma_a = b \log_2 \left(1 + \frac{\rho_a g_a}{\rho_b + r\rho_a \sum_{i \in \mathcal{P}, d_i < g_{\max}} g_i^c} \right) \tag{3}$$

With the defined uploading rate, we can now calculate for the time cost of the offloading process. Supposing player a will offload part of its tasks to the cloud servers, and the data size is s, the time cost of the network transferring can be expressed as

$$t_a^{\text{comm}}(d_a) = \frac{s}{\gamma_a} \tag{4}$$

One more important cost in network communication is the energy consumption, which can be written as

$$e_a^{\text{comm}}(d_a) = \rho_a \frac{s}{\gamma_a} + e_{\text{tail}} \tag{5}$$

Since the commonly deployed cellular networks usually keep the connection even when the communication is finished [19], [35], there will be extra energy consumption in the uploading process. We denote this cost as e_{tail}. In addition, we have omitted the communication cost of control signals in our work, because they are usually in very small size, and have little cost on both energy and time.

According to Eq. (4) and Eq. (5), the total communication cost of player a can be written as

$$C_a^{\text{comm}}(d_a) = \lambda_a^e e_a^{\text{comm}}(d_a) + \lambda_a^t t_a^{\text{comm}}(d_a) \quad , \lambda_a^e + \lambda_a^t = 1 \tag{6}$$

where λ_a^e and λ_a^e are weights for energy consumption and time cost. Each player can set different weight values depending on which cost it cares more.

As we have defined the estimation equation of the communication part, we will continue to introduce the cost estimation of the computing part in the next section.

4.2 Computation Cost Estimation

We denote the set of input image numbers as $\mathcal{M} = \{m_1, ..., m_r\}$, and player a has m_a images in its memory. These images are captured by the depth sensors, and wait to be recognized by this player, one by one. These ten images need to be processed by the deep model for once, forwarding through all the l layers. Although we design a $l = 92$ model in section 3.2, l can be set to any integer, because we want to make a universal algorithm which can be used for CNN models with arbitrary structures. Thus, the computation task is to compute for $m_a * l$ layers. The player needs a decision regarding to which image and to which layer it should calculate for. We define the possible game decisions as a set $\mathcal{G} = \{0, 1, 2, \cdots\}$, which represents the layer number that the player want to calculate. Player a should select a $g \in \mathcal{G}$ as its game decision, making $0 \leq g \leq m_a l$. When g equals 0, player a will conduct no computation on the input images, and directly upload the raw data to the central servers; when $g = m_a l$, player a will perform all the calculation without the help of computing servers, and in this scenario, no image data or intermediate values need to be transferred.

Having the layer-wise information of the cost, namely $\mathcal{E}^{\text{layer}}$ and $\mathcal{T}^{\text{layer}}$, and intermediate data size, namely $\mathcal{S}^{\text{layer}}$, we can define precise estimation models for the computation process. For player a and its game decision d_a, the player will calculate for d_a layers, corresponding to $\lceil \frac{d_a}{l} \rceil$ images. After sending the intermediate values, whose data size is $s^{\text{layer}}_{[d_a \mod (l+1)]} + s_0^{\text{layer}} \lceil \frac{m_a l - d_a}{l} \rceil$, the server will complete the remaining works, i.e. $m_a l - d_a$ layers, corresponding for $\lceil \frac{m_a l - d_a}{l} \rceil$ images. When the computation is complete, the server will output the recognition result, and send it back to the player a. Since this data is very small, we omit it in this research. Therefore, the energy consumption of computation process for player a can be expressed as

$$e_a^{\text{comp}}(d_a) = \sum_{i \in [1, d_a]} e^{\text{layer}}_{(i \mod l)} \tag{7}$$

and the time cost of player a is

$$t_a^{\text{comp}}(d_a) = \sum_{i \in [1, d_a]} t^{\text{layer}}_{\text{player},(i \mod l)} \tag{8}$$

As mentioned above, both the energy and time cost of CNN forwarding are mainly caused by MAC operations. Therefore, there is roughly linear relationship between $\mathcal{E}^{\text{layer}}$ and $\mathcal{T}^{\text{layer}}_{\text{player}}$. Thus,

$$e_a^{\text{layer}} = \sigma t^{\text{layer}}_{\text{player},a} \tag{9}$$

and

$$e_a^{\text{comp}}(d_a) = \sigma t_a^{\text{comp}}(d_a) \tag{10}$$

where σ is the parameter that the energy consumption rate during per time unit.

Then we can formulate the total cost for local computing as

$$C_a^{\text{comp,player}}(d_a) = \lambda_a^e e_a^{\text{comp}}(d_a) + \lambda_a^t t_a^{\text{comp}}(d_a) \quad , \lambda_a^e + \lambda_a^t = 1 \tag{11}$$

Then we need to define a similar estimation model for the server side. As the servers are not limited by energy, they

only cares about the time cost. Therefore, its cost model can be written as

$$t_a^{\text{comp}}(d_a) = \sum_{i \in (d_a, l]} t^{\text{layer}}_{\text{server},(i \mod l)} \tag{12}$$

and the cost value caused by player a in the server side can be expressed as

$$C_a^{\text{comp,server}}(d_a) = \lambda_a^t t^{\text{comp}}_{\text{server},a}(d_a) \tag{13}$$

According to Eq. (11) and Eq. (13), the total computation cost of player a can be written as

$$\begin{aligned} C_a^{\text{comp}}(d_a) &= C_a^{\text{comp,player}}(d_a) + C_a^{\text{comp,server}}(d_a) \\ &= \lambda_a^e e_a^{\text{comp}}(d_a) + \lambda_a^t \left[t^{\text{comp}}_{\text{player},a}(d_a) + t^{\text{comp}}_{\text{server},a} a(d_a) \right] \\ &\quad , \lambda_a^e + \lambda_a^t = 1 \end{aligned} \tag{14}$$

In addition, we can also refine Eq. (4) and Eq. (5) as

$$t_a^{\text{comm}}(d_a) = \frac{s^{\text{layer}}_{[d_a \mod (l+1)]} + s_0^{\text{layer}} \lceil \frac{m_a l - d_a}{l} \rceil}{\gamma_a} \tag{15}$$

and

$$e_a^{\text{comm}}(d_a) = \rho_a \frac{s^{\text{layer}}_{[d_a \mod (l+1)]} + s_0^{\text{layer}} \lceil \frac{m_a l - d_a}{l} \rceil}{\gamma_a} + e_{\text{tail}} \tag{16}$$

4.3 Offloading Algorithm

Based on Eq. (6) and Eq. (14), we can now obtain the overall cost function

$$\begin{aligned} C_a(d_a) &= C_a^{\text{comm}}(d_a) + C_a^{\text{comp}}(d_a) \\ &= \lambda_a^e \left[e_a^{\text{comm}}(d_a) + e_a^{\text{comp}}(d_a) \right] \\ &\quad + \lambda_a^t \left[t_a^{\text{comm}}(d_a) + t^{\text{comp}}_{\text{player},a}(d_a) + t^{\text{comp}}_{\text{server},a} a(d_a) \right] \\ &\quad , \lambda_a^e + \lambda_a^t = 1 \end{aligned} \tag{17}$$

In addition, except the energy consumption and communication cost, the privacy protection is also a key factor which should be seriously considered. Privacy leakage is a common threat to the network security. However, it is not easy to find a strictly secure cloud computing approach, because there are too many possible vulnerabilities which can lead to unauthorized data access. Cloud network, no matter how many safety strategies are deployed, cannot absolutely guarantee the data security. In general, the higher-level features are more difficult to be reversed. Therefore, in order to protect the raw data captured by the mobile robots, which can be owned by personal users, we should limit the directly uploading without any forward calculating. Another important fact is that the central cloud may be overloaded due to the super large data uploaded from numerous robots. As a result, they will be unable to respond to too many CNN computing requests. Thus, we should encourage the players can help with the calculation tasks, and perform at least a small part of the CNN layers. Because of these two reasons, we propose another cost factor to give the local robots to positively take some responsibility. This factor is called local-computing factor C_a^{local}, and it can be expressed as

$$C_a^{\text{local}}(d_a) = \frac{1}{d_a / m_a l} \tag{18}$$

Combining Eq. (17) and (18), we define the final cost function as

$$\begin{aligned} C_a(d_a) = C_a^{\text{comm}}(d_a) + C_a^{\text{comp}}(d_a) + C_a^{\text{local}}(d_a) \\ , \lambda_a^e + \lambda_a^t = 1 \end{aligned} \tag{19}$$

Then using Eq. (8), (10), (15), (16), and (17), we can get the final cost estimation used for the offloading algorithm.

To formulate the offloading game, we suppose a scenario where each player desires to minimize their overall cost, but must follow the instructions made by the offloading controller. In other words, for an arbitrary robot a, it will seek a decision leading to the minimal cost, i.e.,

$$\min_{d \in \mathcal{G}} C_a(d) \tag{20}$$

The controller and the game players will jointly make the decisions regarding how much computation should each of the players take. According to these decisions, the players will correspondingly calculate for some parts of the deployed CNN model, and leave the remaining tasks to the central servers. The game objective is to minimize the overall cost, i.e.,

$$\min_{d_a \in \mathcal{G}} \sum_{a \in \mathcal{P}} C_a(d_a) \tag{21}$$

However, Eq. 20 and Eq. 21 are not equivalent in most cases. The optimal decisions of the game players are usually contradictory, and cannot be performed at the same time. Therefore, each player must publish its desired decision, and contend for the update chance to make the decision acknowledged. But finding an optimal solution to meet Eq. 21 is a NP-hard problem. Therefore, we design a heuristic algorithm to address this problem, as shown in Algorithm 1 and 2.

Algorithm 1: Computation Offloading (Controller)

```
Input: Robot Set 𝒫, Max steps N
/* Initialization.                    */
COUNT = 0;
𝒟 = {0, 0, ··· , 0};
/* Decision-making.                   */
while TRUE do
    COUNT += 1;
    if COUNT >= N then
        SendTerminate(𝒫);
        break;
    SendStart(𝒫);
    𝒰 = ReceiveUpdates(𝒫);
    if 𝒰 = ∅ then
        SendTerminate(𝒫);
        break;
    𝒟 = SelectUpdate(𝒟, 𝒰);
    SendUpdate(𝒟);
```

Algorithm 1 is for the offloading controller and Algorithm 2 is conducted on mobile robots. First, the controller

Algorithm 2: Computation Offloading (Robot)

Input: Deep Model M, Robot ID i
Output: Decision Set $\mathcal{D}$
/* Initialization. */
$\mathcal{D} = \{0, 0, \cdots, 0\}$;
/* Decision-making. */
while *TRUE* **do**
  if *receive TERMINATE* **then**
    return $\mathcal{D}$;
  if *receive START* **then**
    P = MeasureInterference();
    u = ComputeUpdate($\mathcal{D}$,M,P);
    if u != $\mathcal{D}_i$ **then**
      ContendUpdate(u);
    $\mathcal{D}$ = ReceiveUpdate();

will reset the counter and decision set for initialization, and start a new computation offloading process, which is a loop with termination conditions. On the other hand, the robots will also reset their decision sets at the initialization step, and start their running loop, which can only be stopped by the termination commands from the offloading controller. The controller sends the START command to all the robots, and after that, all the robots will first measure the wireless interference P. Then each robot can compute its optimal decision based on Eq. 20. If its current configure is already the same with the calculated optimal decision, it can keep the decision for the following offloading process; otherwise, it should notice the controller that it has a new preferred decision, and contend for the public acceptance, because this new decision may be contradictory with other robots' decisions. After the controller receives all the uploading requests from the robots, the controller will decide which request should be satisfied using a random selection. The selected robot will be granted the right to update its configuration with its own optimal decision, while the others' requests are rejected and can only keep their current configurations. Then the controller will continue to conduct a new round of the loop. The algorithm terminates with two conditions, i.e., exceed the maximum loop numbers, or no robots contend for the updating chances.

The proposed algorithm can well solve the offloading problem, and find the optimal solution to meet Eq. (21). We will demonstrate this in the experiment section with extensive numerical analysis, including its ability to decrease the overall cost and its coverage speed when applied with a large number of robots.

5 Performance Evaluation

In this section, we conducted a series of experiments to demonstrate the proposed method, mainly including two aspects, i.e., the recognition experiment, and the computation offloading simulation.

The experimental settings are shown in Table. 2 for better reproduction and evaluation of the proposed method. In the experimental scenarios, every wireless base station can cover a range of about $200m$, and we respectively deploy one, two and three stations in the three scenarios shown in Fig. 4. The points in the figure are the mobile robots, and they will connect to the closest stations for better communication quality. The different colors and shapes can represent which station the robots select.

TABLE 2
The Settings of Offloading Experiments

Item	Value
Station Cover Range	220 m
Station Number	$\{1, 2, 3\}$
Bandwidth b	1.25 MHz
Path Loss Factor α	4
Background Noise ρ_b	-100 dBm
Image Number m	$[1, 20]$
Layer Number l	92
Energy/Time Ratio σ	20

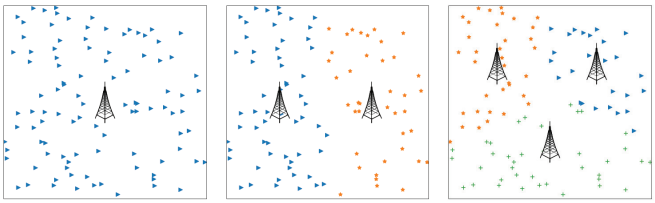


Fig. 4. Device Locations. Sub-figure (a)(b)(c) are three different sets of the offloading experiments. We respectively adopt single, double and triple wireless base stations in these three experiments. The points in the figure are the randomly-located robots. They connect to the closest stations for better communication rate. We use different colors and shapes to represent their belongings.

5.1 Recognition Experiment

We jointly adopted several different datasets in this work, including the RGBD Object dataset [36], Bigbird dataset [37], and A Large Dataset of Object Scans [38]. 300 categories of objects are selected, and each of them has 300 instances, in which 250 instances were adopted as the training materials, and the remaining 50 instances were used as the test set.

In this experiment, we mainly test the recognition performance of the proposed system. Three CNN models were selected, namely, the simplified VTCN model, a 30-layer CNN model, and the proposed 92-layer CNN model. The VTCN model was extracted from our last work, and we simplified it into a six-layer model. The 30-layer model is a small version of our 92-layer model, with all necessary layers and parameters. We conducted this experiment to obtain the comprehensive measurement of their recognition performance, including the precision and the consumption.

As shown in Fig. 5a, with the increase of the model complexity, the recognition accuracy also improved, which is consistent with the recent findings in deep learning field. Therefore, it is a good way to improve the model complexity for better recognition precision. However, the problem is, that the consumption also increased quickly with the model complexity, which is also our focus in this research.

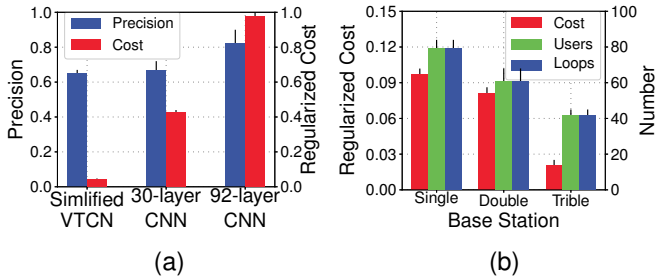


Fig. 5. Experimental results. (a)The comparison experiment of the CNN recognition performance. The selected methods include a six-layer VTCN model, a 30-layer CNN model and the 92-layer model proposed in this paper. The left y-axis is the precision, and the right y-axis is their energy consumption. (b)The comparison results of the different settings. Single, double, triple wireless base stations are respectively adopted in these three experiments. The left y-axis represents the system cost, and the right y-axis represents the number of benefited robots.

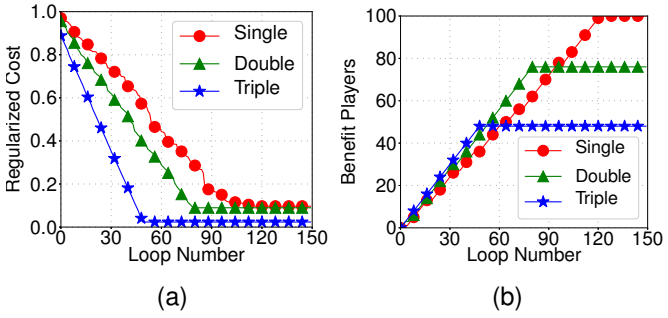


Fig. 6. Algorithm running results. (a)The converge curve of the proposed offloading algorithm. The x-axis is the loop number and the y-axis is the overall cost. (b)The curve of benefited robots. The x-axis is the loop number and the y-axis is total number of robots who decrease their own costs with the propose offloading algorithm.

5.2 Offloading Experiment

In this section, we test the offloading performance of the proposed method. Fig. 5b compares the total system cost, benefited robot number, and the converge steps in these three different scenarios. Generally, the more stations there were, the higher the communication quality was. In this experiment, the increase of the station number brought a significant decrease on all of the three measurements.

Fig. 6a and Fig. 6b are some results recorded in single station environment, where 100 mobile robots exist. The former one shows the converge curve of our computation offloading method. We can see that it is a monotonically decreasing curve, and quickly reach that no robot can get more benefits. Fig. 6b gives the change of benefited robot number, it can be seen that more and more robots obtained a decreased total consumption, with the execution of the proposed offloading method. The final number of single station scenario reached 100%, which means all of the robots got benefits from the offloading algorithm. In other two scenarios, a large number of the robots were benefited from the offloading process.

Fig. 7 presents the change curve of the overall cost with the increase of the robot number. One obvious influence caused by the increase of the robot density was the severe deterioration of the communication quality. Therefore, when

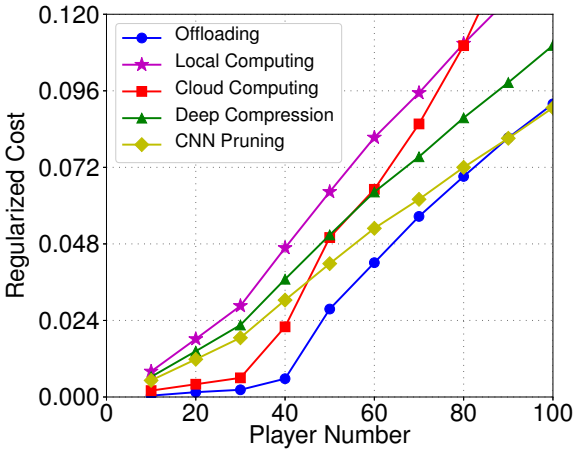


Fig. 7. The relationship between the robot number and the overall cost. The x-axis is the total number of robots, and the y-axis is the total system cost. The blue line is the proposed offloading method, the magenta line represents conducting all computational tasks locally, and the red line is to directly transfer the raw data to the cloud servers. As a comparison, we also present two CNN energy-saving methods as the green line and the yellow line.

there exists a small number of robots, leaving all computation tasks to the server side is a good solution to decrease the system consumption. However, with the increase of robot number, the communication cost was significantly increased, and the cost of cloud computing was correspondingly increased. As a comparison, the cost of local computing followed a near-linear relationship with the total computation tasks. Although it costs more when the communication condition is good, it is a better selection when the robot density is too high. The best performance was from the proposed method, and outperformed other two approaches in all the conditions. In addition, we compare our offloading method with two energy-saving approaches, namely, the deep compression and the CNN pruning. Their main concept is to decrease the computation cost of the CNN models through some pruning processes. We can see that these two methods show some advantages against the original local computing method, but our offloading method outperforms them in most cases. Even when the robot density is very large, our method can still have a similar performance.

6 Conclusion

In this paper, we proposed a novel CNN model for 3D scene understanding tasks. For sustainability, we formulated the computation and communication model of the proposed network, and designed an offloading strategy to balance the local computing and cloud offloading for better energy-saving performance. The experimental results proved that the proposed approach in this paper is efficient and effective.

In the future, we will continue to optimize the offloading strategy, refining the cost estimations, communication models, which are not very satisfactory in some scenarios, and the control algorithm for better performance and adaptability.

ACKNOWLEDGMENT

This work is partially supported by JSPS KAKENHI Grant Number JP16K00117, JP15K15976, and KDDI Foundation. Mianxiong Dong is the corresponding author.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with microsoft kinect sensor: A review," *IEEE Transactions on Cybernetics*, vol. 43, no. 5, pp. 1318–1334, Oct 2013.

[3] C. Chen, R. Jafari, and N. Kehtarnavaz, "A survey of depth and inertial sensor fusion for human action recognition," *Multimedia Tools and Applications*, vol. 76, no. 3, pp. 4405–4425, Feb 2017.

[4] L. Li, K. Ota, M. Dong, and W. Borjigin, "Eyes in the dark: Distributed scene understanding for disaster management," *IEEE Transactions on Parallel and Distributed Systems*, 2017, doi: 10.1109/TPDS.2017.2740294.

[5] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 2377–2385.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[7] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng, "Convolutional-recursive deep learning for 3d object classification," in *Advances in Neural Information Processing Systems*, 2012, pp. 656–664.

[8] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[9] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust rgb-d object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 681–687.

[10] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 922–928.

[11] K. Ota, M. Dong, J. Gui, and A. Liu, "Quoin: Incentive mechanisms for crowd sensing networks," *IEEE Network Magazine*, vol. PP, no. 99, pp. 1–1, 2018, doi:10.1109/MNET.2017.1500151.

[12] H. Li, K. Ota, M. Dong, and M. Guo, "Mobile crowdsensing in software defined opportunistic networks," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 140–145, 2017.

[13] H. Li, K. Ota, M. Dong, and H. H. Chen, "Efficient energy transport in 60 ghz for wireless industrial sensor networks," *IEEE Wireless Communications*, vol. 24, no. 5, pp. 143–149, October 2017.

[14] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *2012 Proceedings IEEE INFOCOM*, March 2012, pp. 945–953.

[15] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," in *2012 Proceedings IEEE INFOCOM*, March 2012, pp. 2716–2720.

[16] O. MuÃśoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 10, pp. 4738–4755, Oct 2015.

[17] J. Kwak, Y. Kim, J. Lee, and S. Chong, "Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 12, pp. 2510–2523, Dec 2015.

[18] L. Yang, J. Cao, S. Tang, T. Li, and A. T. S. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," in *2012 IEEE Fifth International Conference on Cloud Computing*, June 2012, pp. 794–802.

[19] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, October 2016.

[20] K. Gai, M. Qiu, H. Zhao, L. Tao, and Z. Zong, "Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing," *Journal of Network and Computer Applications*, vol. 59, pp. 46 – 54, 2016.

[21] K. Gai, M. Qiu, and H. Zhao, "Energy-aware task assignment for mobile cyber-enabled applications in heterogeneous cloud computing," *J. Parallel Distrib. Comput.*, vol. 111, no. C, pp. 126–135, Jan. 2018. [Online]. Available: https://doi.org/10.1016/j.jpdc.2017.08.001

[22] X. Tao, K. Ota, M. Dong, H. Qi, and K. Li, "Performance guaranteed computation offloading for mobile-edge cloud computing," *IEEE Wireless Communications Letters*, vol. 6, no. 6, pp. 774–777, Dec 2017.

[23] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient dnns," in *Advances In Neural Information Processing Systems (NIPS)*, 2016, pp. 1379–1387.

[24] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *International Conference on Learning Representations (ICLR)*, 2016.

[25] S. Srinivas and R. V. Babu, "Data-free parameter pruning for deep neural networks," in *BMVC*, 2015.

[26] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," *arXiv preprint arXiv:1607.03250*, 2016.

[27] Q. Zhang, L. T. Yang, and Z. Chen, "Privacy preserving deep computation model on cloud for big data feature learning," *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1351–1362, May 2016.

[28] D. Das, S. Avancha, D. Mudigere, K. Vaidynathan, S. Sridharan, D. Kalamkar, B. Kaul, and P. Dubey, "Distributed deep learning using synchronous stochastic gradient descent," *arXiv preprint arXiv:1602.06709*, 2016.

[29] X. Ran, H. Chen, Z. Liu, and J. Chen, "Delivering deep learning to mobile devices via offloading," in *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*, ser. VR/AR Network '17. New York, NY, USA: ACM, 2017, pp. 42–47.

[30] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1–9.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems*, ser. NIPS'12. USA: Curran Associates Inc., 2012, pp. 1097–1105.

[32] Y. H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, June 2016, pp. 367–379.

[33] Y. H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan 2017.

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[35] W. Hu and G. Cao, "Quality-aware traffic offloading in wireless networks," *IEEE Transactions on Mobile Computing*, vol. PP, no. 99, pp. 1–1, 2017.

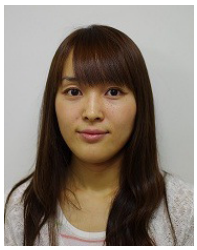
[36] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 1817–1824.

[37] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel, "Bigbird: A large-scale 3d database of object instances," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 509–516.

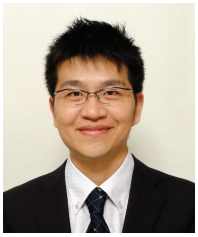
[38] S. Choi, Q.-Y. Zhou, S. Miller, and V. Koltun, "A large dataset of object scans," *arXiv:1602.02481*, 2016.



Liangzhi Li received the B.Sc and M.Eng degrees in Computer Science from South China University of Technology (SCUT), China, in 2012 and 2016, respectively. He is currently pursuing the Ph.D. degree in Electrical Engineering at Muroran Institute of Technology, Japan. His main fields of research interest include machine learning, big data, and robotics. He has received the best paper award from FCST 2017.



Kaoru Ota was born in Aizu-Wakamatsu, Japan. She received M.S. degree in Computer Science from Oklahoma State University, USA in 2008, B.S. and Ph.D. degrees in Computer Science and Engineering from The University of Aizu, Japan in 2006, 2012, respectively. She is currently an Assistant Professor with Department of Information and Electronic Engineering, Muroran Institute of Technology, Japan. From March 2010 to March 2011, she was a visiting scholar at University of Waterloo, Canada. Also she was a Japan Society of the Promotion of Science (JSPS) research fellow with Kato-Nishiyama Lab at Graduate School of Information Sciences at Tohoku University, Japan from April 2012 to April 2013. Her research interests include Wireless Networks, Cloud Computing, and Cyber-physical Systems. Dr. Ota has received best paper awards from ICA3PP 2014, GPC 2015, IEEE DASC 2015, IEEE VTC 2016-Fall, FCST 2017 and 2017 IET Communications Premium Award. She is an editor of IEEE Transactions on Vehicular Technology (TVT), IEEE Communications Letters, Peer-to-Peer Networking and Applications (Springer), Ad Hoc & Sensor Wireless Networks, International Journal of Embedded Systems (Inderscience) and Smart Technologies for Emergency Response & Disaster Management (IGI Global), as well as a guest editor of ACM Transactions on Multimedia Computing, Communications and Applications (leading), IEEE Internet of Things Journal, IEEE Access, IEEE Communications Magazine, IEEE Network, IEEE Wireless Communications (2015), IEICE Transactions on Information and Systems (2014), and Ad Hoc & Sensor Wireless Networks (Old City Publishing) (2014). She is the recipient of IEEE TCSC Early Career Award 2017.



Mianxiong Dong received B.S., M.S. and Ph.D. in Computer Science and Engineering from The University of Aizu, Japan. He is currently an Associate Professor in the Department of Information and Electronic Engineering at the Muroran Institute of Technology, Japan. He was a JSPS Research Fellow with School of Computer Science and Engineering, The University of Aizu, Japan and was a visiting scholar with BBCR group at University of Waterloo, Canada supported by JSPS Excellent Young Researcher Overseas Visit Program from April 2010 to August 2011. Dr. Dong was selected as a Foreigner Research Fellow (a total of 3 recipients all over Japan) by NEC C&C Foundation in 2011. His research interests include Wireless Networks, Cloud Computing, and Cyber-physical Systems. He has received best paper awards from IEEE HPCC 2008, IEEE ICESS 2008, ICA3PP 2014, GPC 2015, IEEE DASC 2015, IEEE VTC 2016-Fall, FCST 2017 and 2017 IET Communications Premium Award. Dr. Dong serves as an Editor for IEEE Transactions on Green Communications and Networking (TGCN), IEEE Communications Surveys and Tutorials, IEEE Network, IEEE Wireless Communications Letters, IEEE Cloud Computing, IEEE Access, as well as a leading guest editor for ACM Transactions on Multimedia Computing, Communications and Applications (TOMM), IEEE Transactions on Emerging Topics in Computing (TETC), IEEE Transactions on Computational Social Systems (TCSS). He has been serving as the Vice Chair of IEEE Communications Society Asia/Pacific Region Meetings and Conference Committee, Leading Symposium Chair of IEEE ICC 2019, Student Travel Grants Chair of IEEE GLOBECOM 2019, and Symposium Chair of IEEE GLOBECOM 2016, 2017. He is the recipient of IEEE TCSC Early Career Award 2016, IEEE SCSTC Outstanding Young Researcher Award 2017 and The 12th IEEE ComSoc Asia-Pacific Young Researcher Award 2017.