applied sciences

MDPI

Article

Deep Learning Entrusted to Fog Nodes (DLEFN) Based Smart Agriculture

Kyuchang Lee, Bhagya Nathali Silva and Kijun Han *

School of Computer Science and Engineering, Kyungpook National University, Daegu 41566, Korea; skynsoul@knu.ac.kr (K.L.); nathalis@netopia.knu.ac.kr (B.N.S.)

* Correspondence: kjhan@knu.ac.kr

Received: 4 February 2020; Accepted: 19 February 2020; Published: 24 February 2020

Abstract: Colossal amounts of unstructured multimedia data are generated in the modern Internet of Things (IoT) environment. Nowadays, deep learning (DL) techniques are utilized to extract useful information from the data that are generated constantly. Nevertheless, integrating DL methods with IoT devices is a challenging issue due to their restricted computational capacity. Although cloud computing solves this issue, it has some problems such as service delay and network congestion. Hence, fog computing has emerged as a breakthrough way to solve the problems of using cloud computing. In this article, we propose a strategy that assigns a portion of the DL layers to fog nodes in a fog-computing-based smart agriculture environment. The proposed deep learning entrusted to fog nodes (DLEFN) algorithm decides the optimal layers of DL model to execute on each fog node, considering their available computing capacity and bandwidth. The DLEFN individually calculates the optimal layers for each fog node with dissimilar computational capacities and bandwidth. In a similar experimental environment, comparison results clearly showed that proposed method accommodated more DL application than other existing assignment methods and utilized resources efficiently while reducing network congestion and processing burden on the cloud.

Keywords: fog computing; deep learning; layer definition algorithm; smart agriculture

1. Introduction

In rural areas, the increasing age of populations and reduction in quantity will lead to weakness of economic power due to labor shortage. Smart agriculture technologies are already being used in many countries as an approach, using information and communication technology (ICT) to effectively solve rural problems such as the above-mentioned labor shortage [1,2]. Recently, smart agriculture has attempted and implemented full automation incorporating other trending technologies such as big data, Internet of Things (IoT), and artificial intelligence (AI) to improve farmers' quality of life (QoL) [3].

AI has been extensively studied for decades and has become popular today, along with other high-end technologies [4]. AI technologies are used to extract valuable data from various data generated by IoT devices. Deep learning (DL), which is an AI technology, is effective in analyzing huge multimedia data such as images and videos. DL technology can be used to identify health conditions by analyzing the movement of livestock or to optimize the timing of harvests by analyzing the shape of agricultural products [5]. Thus, the proper utilization of DL in smart agriculture significantly improve farmers' economic power [6].

Fog computing (FC) is another recent technology that has improved quality of service (QoS) for modern ICT in the 4th industrial revolution. Unlike cloud computing, which processes all data at a central server, fog nodes distributed in FC support the processing on a central server [7]. In modern environments where huge volumes of multimedia data are continuously generated from IoT

media devices such as CCTV [8], processing all data on a central server may bring on low QoS, such as response delay due to burden of the server and limitations of the network. In FC, intermediate servers called fog nodes preprocess some of data received from devices and share the burden of the central server.

In DL processes, valuable results are obtained after incoming data traverse through many layers, since a DL model is composed of multiple layers to analyze features. Processing at each layer extracts features and generates data for delivery to the next layer; the size of the generated data is thus significantly reduced compared with the incoming size. An architecture of DL models is appropriate for a FC system, since fog nodes can handle some layers that process incoming data and the cloud server can then handle residual layers to process the data transferred from fog nodes. Fog nodes reduce the data volume by dealing with the authorized DL layers. Therefore, DL operation on FC can reduce overall network traffic and computational overhead on a cloud server, thereby minimizing response delays and allowing more DL operations to be accommodated.

Herein, we propose a deep learning entrusted to fog nodes (DLEFN) algorithm that decides the optimal number of DL layers that can be handled by each fog node. The proposed DLEFN aims to improve the availability and utilization of fog nodes distributed in an FC environment and to accommodate many applications effectively. The optimal layers are decided by taking into account the available computational capacity and bandwidth at each fog node. DL processing at fog nodes reduces the data volume sent to the cloud, which therefore reduces computational overhead at the cloud and network. The available resource capacities at each fog node commonly have the possibility of being different due to new device installation, execution of extra tasks, received data size, and network facilities. Considering an environment made up of fog nodes with differences in performance, the DLEFN defines handling layers for DL tasks at each fog node individually. We experimented with and compared the proposed DLEFN algorithm and other methods such as simple gateway and static and dynamic layer decisions. The results showed considerable improvement associated with the DLEFN compared to existing schemes in the number of accommodated DL applications and resource utilization at fog nodes. Therefore, we can clearly assert that the proposed method effectively accommodates more DL applications at fog nodes than other methods.

The rest of the paper is divided into the following sections and subsections. Some related research on the use of DL in cloud and FC environments is explained in Section 2. The utilization of DL-based smart agriculture is presented in Section 3, and the overview of DL grafted onto an FC environment and the proposed optimal DL layer definition methods are described in Section 4. Section 5 presents the analysis and discussion of the algorithm's performance. Finally, Section 6 presents the conclusions of the work.

2. Literature Review

DL is a kind of artificial neural network (ANN) consisting of many layers that are used to process input data. ANN is the core technology of DL, inspired by the biological neural network of the human brain. An artificial neuron (AN) abstracts the functions of a biological neuron, which consists of a dendrite, soma, and axon. In a neuron, the dendrite performs the function of receiving input signals through electrical signals; the soma sums up the inputs received through the dendrite; and the axon output values are calculated by soma [9]. Figure 1 illustrates a structure of a biological and an artificial neuron. An AN has a limited number of inputs with weights related to them, and an activation function. The output of the AN is the result of the activation function applied to the weighted sum of inputs. ANs are connected with each other to form ANNs that have multiple layers, including an input layer, and output layer, and at least one more hidden layer [9]. Figure 2 depicts an example of a fully connected, multiple-layered ANN with n hidden layers. Each layer of an ANN consists of ANs, and each AN is connected to all the ANs in the next layer. Data move from the input layer through the hidden layers to the output layer.

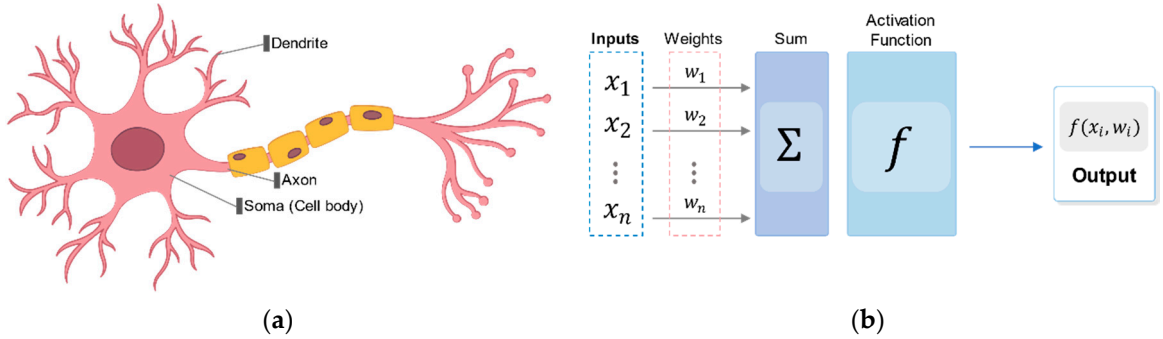


Figure 1. Structures of biological and artificial neurons. (**a**) Biological neuron; (**b**) artificial neuron.

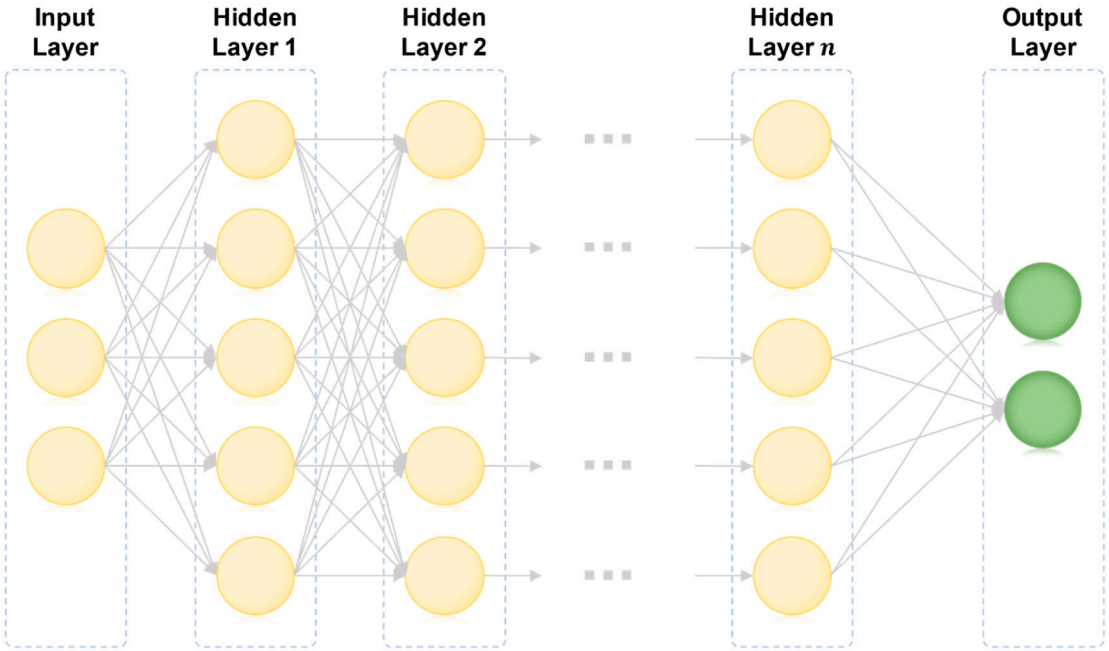


Figure 2. Example of a fully connected, multiple-layered artificial neural network.

DL is able to handle colossal multimedia data, and DL techniques are able to perform highly accurate object awareness analysis from video and image data [10]. IoT-based application developers are interested in the benefits of DL, since modern IoT devices constantly generate multimedia data that must be analyzes in order to obtain useful information. Smart agriculture also utilizes many IoT multimedia devices and data generated from them to support smart services such as facility automation.

Nonetheless, IoT-based data collection hardware is commonly inadequate to independently perform DL due to restricted energy and computational capacity [11]. Accordingly, several studies have been performed to support DL on IoT devices. Cloud computing is a common approach to address low performance issues in IoT devices. However, the continued generation of colossal volumes of data leads to network congestion and focused overhead of the central server in cloud environment [12]. To overcome this issue, fog nodes in FC process part or all of the data before sending it to the cloud. Thus, FC mechanisms considerably reduce data traffic and burden on the cloud. Keeping these advantages in mind, some studies have attempted to distribute DL overhead by applying FC mechanisms.

Lane et al. [13] suggested DeepEar with DeepX as novel acceleration schemes to assist dissimilar applications with DL on mobile systems on chip. Their experimental results showed that a mobile device with high performance can deal with a portion of DL processing. Alsheikh et al. [14] suggested a combination of DL and the Apache Spark application for analyzing and learning data at the same time in a cloud environment. Their mobile devices performed DL operations to analyze data while the

Apache Spark learned data on the cloud simultaneously. In this platform with its two-layer structure, mobile devices were able to capably handle a portion of the work on the cloud.

Gupta et al. [15] suggested iFogSim to model and simulate an IoT and FC environment. The scheme experimented with average latency and execution time for controlling a loop of intelligent surveillance systems in a cloud and FC environment. The results showed that system control was more competent in FC than cloud computing with regards to energy and execution time. Liu et al. [16] suggested a food classification scheme utilizing DL and FC. The scheme reduced the response time and power consumption of DL applications by sharing works on the FC.

Lavassani et al. [17] suggested a new distributed learning model for sensor devices. The proposed work simulated the data stream by making a connection with the fog nodes. In order to save energy and communicate as few packets as possible, the updated parameters of the learned model at the sensor devices were communicated in longer time intervals to the FC system. This framework showed that a combination of fog and cloud computing with distributed data modeling at the sensor devices could be advantageous for IoT applications. Li et al. [18] suggested a scheduling method that offloads a portion of the DL layers to edge servers. Further, the suggested edge computing method attempted to define the appropriate layers of the edges for executing multiple DL tasks. The DL layers of edges were decided based on the largest volumes of incoming data for each task requiring DL, and then set up to all edge servers.

Despite the improvements reported in the literature, there is still much room for research into the implementation of DL in FC-based IoT environments. In general, the available resources of fog node devices are uneven in the real world for many reasons, such as incoming data size, establishment time, and network infrastructure. Nonetheless, when all fog nodes work their best according to each available resource, the burdens of cloud and network traffic are reduced, and more DL applications can be accommodated in an FC environment. Hence, we intended to treat one of the main challenges for implementing DL in FC systems, namely the decision of optimal layers despite uneven available resource in fog nodes.

3. Deep Learning in Smart Agriculture

Current smart agriculture systems consist of various ICT devices such as a variety of sensors, high-definition camera devices like CCTVs, autonomous drones, and voice recognition equipment. Such devices generate colossal volumes of data that include unstructured multimedia data such as videos, images, and sounds as well as structured numerical values such as temperature, humidity, illumination, and air condition [1]. In smart agriculture, the collected data are helpful in making optimal decisions for facility automation, such as optimal temperature and humidity control, and automatic feed supply. Commonly, multimedia data have a larger size than structured numeric data. Moreover, complicated algorithms are required to obtain the appropriate information from large media data. Although generated multimedia data are compressed and encoded, audio data in MP3 format by CBR (constant bit rate) have a bit rate of 320 kbps, and HD video (720p, 60 fps) and FHD video (1080p, 60 fps) have bit rates of 3500 kbps and 5500 kbps respectively. In a modern IoT environment, such colossal multimedia data are flowing into a network continuously. Thus, sufficient resources like computing power and bandwidth are essential to execute DL tasks such as context awareness from video data in a real-time environment.

In order to improve the QoL of farmers, many applications in smart agriculture utilize DL tasks for a variety of tasks, such as commercial value checking for agricultural products, condition management for domestic animals, population estimation, and security [2,3]. These tasks involve the analysis of multimedia data generated by many devices and the extraction of meaningful information to use as a basis for executing processes of applications. The level of convenience and benefit provided to users is determined by the accuracy of the extracted information. Accordingly, many studies have made efforts to improve the quality of the information obtained through DL processing, such as object recognition, context awareness, and character reading. Despite the steady improvement in

accuracy among researched DL technologies with similar purposes, handling DL on IoT-based devices with limited energy and computing power remains an important challenge [11]. To overcome these limitations, a universal approach grafting DL technology onto a cloud service is shown in Figure 3. An application collects media data from devices which it can control, and requests analysis of the required information from a cloud server with enough computing power. A DL algorithm for application on the cloud extracts meaningful information from the received data. Subsequently, the information returned from the cloud is used to execute defined functions of the application. Devices transfer data to either a cloud or a gateway according to network infrastructure. If unable to connect to the cloud, devices transfer data to a nearby gateway that can forward it to the cloud.

Figure 3. Example of deep learning (DL) service grafted onto cloud computing.

Nevertheless, in a cloud-based DL environment, there are two issues that require focus, i.e., (1) network congestion owing to huge data transfer and (2) intensive burden of cloud servers [19]. Even in the cases of intermediate gateways being used to forward data, they demand a very high bandwidth to correspond to all data gathered from related devices. Additionally, cloud servers with high performance also have limitations in terms of simultaneous execution of many DL tasks with huge data [20]. Namely, these two issues raise the possibility of response relay. However, it is worth noting that even a small delay can cause economic losses in smart farms dealing with fields of automation such as production, sales, growth, management, and security.

4. Proposed Solution

4.1. Deep Learning Services on Fog Nodes

The purpose of this work was to minimize response delay and process more DL tasks for smart agriculture applications. The proposed DL model is composed of multiple layers, and each layer deals with incoming data from the previous layer to extract features and generate an intermediate result to be delivered to the next layer. The lowest layer handles pure data and the highest layer outputs valuable information. Each DL layer reduces the data volume to be delivered to the next layer; the

output data volume of each layer is smaller than the input data volume of that layer. Thus, a greater number of DL layers can output more valuable information with smaller size.

Figure 4 depicts the DL model's implementation in an FC environment. A cloud server entrusts some layers of the DL model to fog nodes and take care of the residual layers. Each device is aware of information of a nearby fog node and transfers captured data to the respective fog node at the request of applications. Receiving pure data from devices, a fog node executes defined layers of the DL model corresponding to a DL task application and outputs intermediate results. Upon finishing the process of DL layers at the fog node, the intermediate results are delivered to the central cloud server, which executes residual DL layers to return final information. Since the cloud takes data already processed at fog nodes, the data volumes handled by the cloud are smaller than the unprocessed original data. Such reduction of data volumes leads to smaller loads for transmission throughout the entire network, thus preventing congestion on the network and lowering the computational burden on the cloud server. Consequently, the FC system can accommodate more DL tasks at the same time. Additionally, DL layers divided into fog nodes and a cloud server have a security advantage, since inferring original data from a partially processed intermediate result is difficult in the event of a malicious threat that steals data.

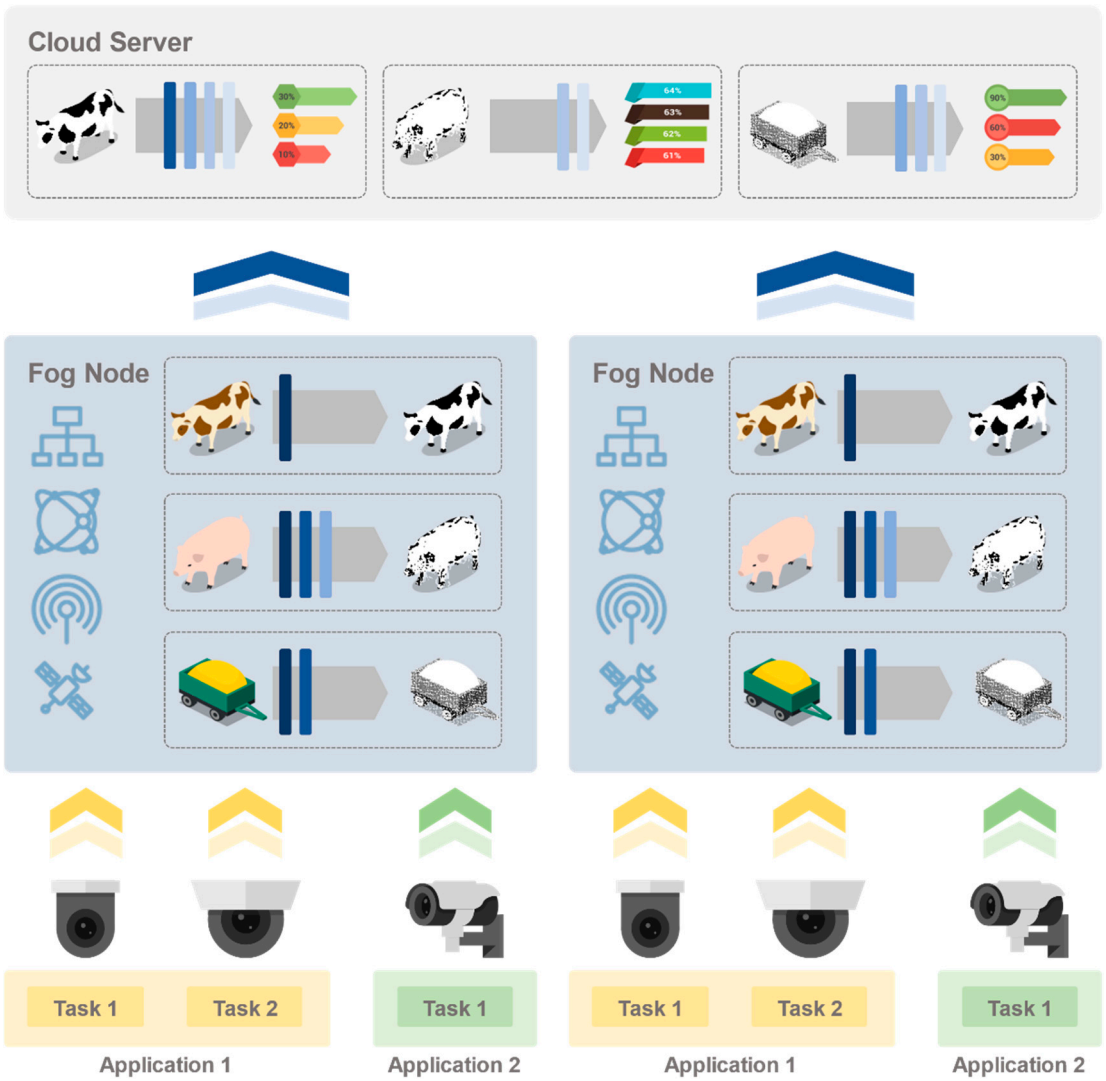


Figure 4. Example of a DL service grafted onto fog computing.

In this paper, the word application signifies a piece of software consisting of DL tasks that utilize data from several devices to provide convenience. Authorized users are permitted to access and use functions of this application. The cloud manages the DL operational algorithm for each DL task. In a cloud environment, the central cloud deals with all executions of DL. In an FC environment, by contrast, fog nodes execute fragmentary layers and then the cloud handles the residual layers. Figure 5 depicts an overview of a DL service for an application in FC environment. The cloud has pretrained, full DL models for tasks of the application and entrusts some appropriate layers of each model to fog nodes according to their available resources.

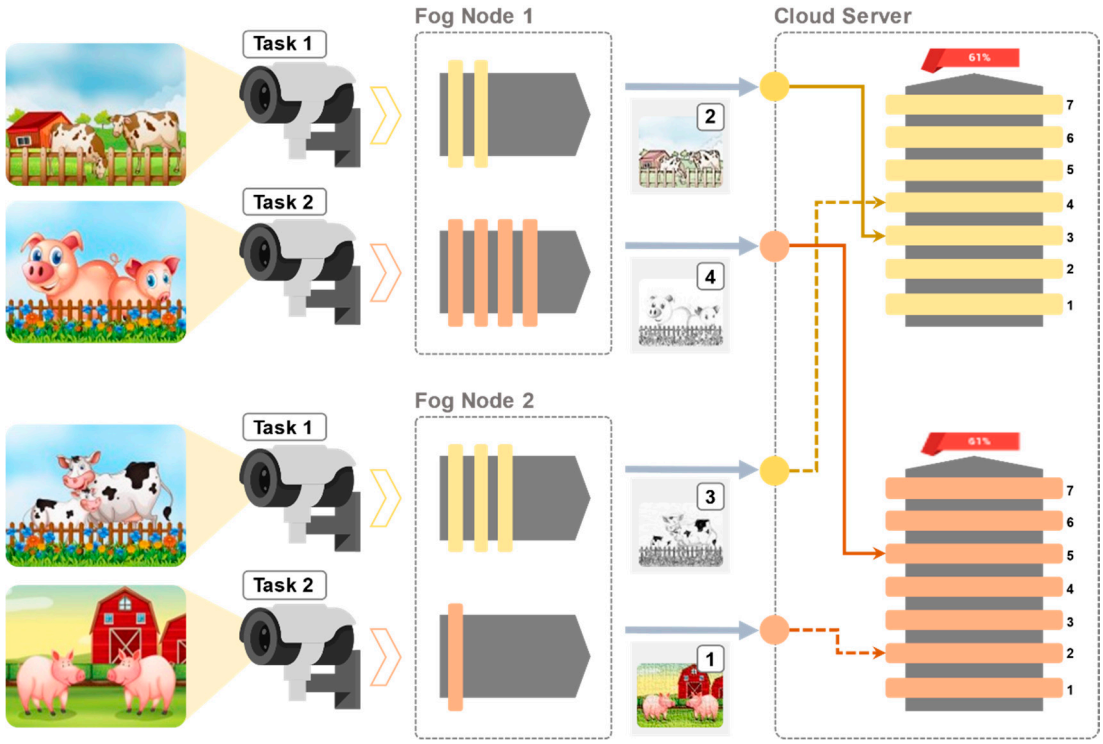


Figure 5. Overview of DL services for two tasks of an application in an FC environment.

We assume that the cloud periodically collects learning information from all fog nodes and disseminates the improved feature map based on the aggregated information. When either a human or an auto-program as a user requires the use of a DL task to analyze multimedia data generated from devices and obtain meaningful information, the data are transferred to a nearby fog node. Upon arrival of the data, the fog node executes the defined DL layer and delivers intermediate results with reference information about node ID, task, and layer to the cloud. The cloud checks the reference information received from fog node and handles intermediate results via execution of residual layers, and finally returns the results to the requested user.

In order to achieve maximum efficiency of the DL model grafted onto an FC-based smart agriculture environment, it is necessary to assign the optimal layers of the DL model for each task to each fog node, in order that DL service may be supported for maximum applications within a given resource. As mentioned earlier, the harder a fog node works, the less data is generated; the more layers are assigned to fog nodes, the smaller the data volumes delivered to the cloud via the network, eventually reducing network congestion and computational burden on the cloud.

Nonetheless, fog nodes require computing resources to handle DL layers, as well as generally having lower performance than the cloud server. Since fog nodes quickly arrive at the limits of their performance if required to handle many DL layers without a plan, the number of DL tasks that FC can process simultaneously is restricted despite the small computing load on the cloud. Thus, the available resources of each fog node are an important parameter to be considered in developing strategies to

maximize the DL applications that can be accommodated. The available resources and bandwidth for each fog node are influenced by infrastructures and incoming data volumes. Therefore, in the next section, we propose an algorithmic strategy named DLEFN to accommodate a maximum number of DL applications and ensure QoS while maximizing resource utilization of fog nodes.

4.2. Proposed DLEFN Algorithm

The proposed DLEFN algorithm defines the optimal DL layers for fog nodes in order to accommodate the maximum number of applications requiring DL services during smart agriculture applications' processing in an FC environment. The proposed DLEFN uses two procedures to accommodate a new DL application, as shown in Figure 6. When a new application with DL tasks is introduced to a given environment, the first procedure of the DLEFN focuses on defining the maximum number of DL layers for each fog node within its available resources. However, if a fog node has insufficient resources to accept even the defined minimum layers, it reduces the number of layers allowed for other DL tasks to guarantee a minimum resource capacity for an attempted task. When even one of the fog nodes has problems caused by insufficient computing capacity and bandwidth, the other procedure of the DLEFN attempts to give opportunity for applications that cause lower burden by comparing the allowed and new applications.

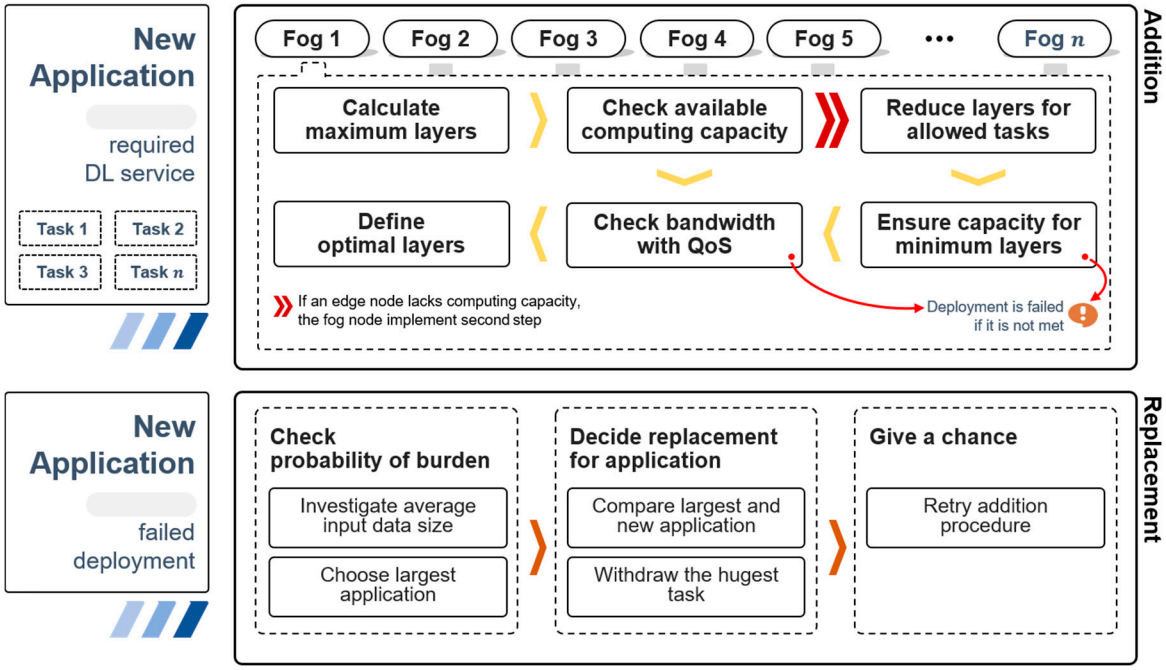


Figure 6. Operational overview of proposed deep learning entrusted to fog nodes (DLEFN) algorithm in DL task deployment architecture.

The components of DL services grafted onto an FC system are as follows. A set of fog nodes dispersed in an area is denoted by F, and a fog node that belongs to the set is denoted by i for identification. The given processing capacity on a fog node i is represented by c_i^{max}, which also describes maximum capacity to handle tasks at the same time. The allowed maximum bandwidth for transmission from fog node i to the cloud is represented by b_i^{max}. Each fog node has different processing capacity and network bandwidth according to time or environment of installation, like a real environment.

Applications consist of one or more tasks requiring DL service and used by multiple users in various places. A set of DL applications accommodated in FC-based smart agriculture is denoted by A, and an application that belongs to set A is defined as a. A new application that does not belong the set

is denoted as n. A set of tasks for an application is denoted by T. Accordingly, application a and n have task sets T_a and T_n, respectively, and t denotes a task that belongs to T.

L_t^{total} indicates the total number of DL layers for task t. l_t^k and r_t^k, up to the k th layer $\left(1 \le k \le L_t^{total}\right)$ for task t, respectively denote the average computing loads per data unit and the average reduction ratio for incoming data. Accordingly, d_t^i denotes the average incoming data size per unit time for task t at fog node i; the required capacity corresponding loads for handling it until the k th layer is calculated by $l_t^k \times d_t^i$; and the data size after the k th layer is calculated by $r_t^k \times d_t^i$. Furthermore, we consider transmission delay to guarantee QoS. The allowable maximum transfer delay for task t is defined by Q_t.

As stated above, the DLEFN consists of two procedures to enable DL service for a new application. The first procedure attempts to define up to the largest layer to each fog node within its individual available computing capacity and bandwidth.

$$L_{tn}^i = \max\left\{k \left| \begin{array}{l} l_{tn}^k \times d_{tn}^i \le c_i^{able}, \\ r_{tn}^k \times \frac{d_{tn}^i}{Q_{tn}} \le b_i^{able}, \\ L_i^{min} \le k \le L_{tn}^{total} \end{array} \right.\right\} \quad s.t., \quad tn \in T_n \tag{1}$$

L_{tn}^i is the number of DL layers that handle data for task tn of new application n at a fog node i. After estimating the expected average data size d_{tn}^i based on information on the volume of data generating from devices connected to a fog node i related to task tn, the first procedure calculates the computational capacity required to handle the incoming data and the bandwidth that ensures QoS when transferring the intermediate result to cloud, and then decides the maximum number of layers that the fog node i can execute within usable resources. c_i^{able} and b_i^{able} indicate available computing capacity and bandwidth on fog node i, respectively. L_i^{min} signifies the predefined compulsory minimum layer for fog node i. In other words, fog node i should execute defined minimum layer or more for DL tasks. Defining maximum layers is tried for all tasks belonging to the new application in each fog node individually.

During the above process for all tasks of a new application, if available computational capacity in fog node i is insufficient, the fog node reduces the already defined layers of tasks of other allowed and new applications. This work ensures minimum capacity to handle the task at the fog node. The work repeatedly cancels a layer that causes minimum loads among the highest layer for each task assigned to fog node i until capacity is guaranteed.

$$L_{ta}^i = L_{ta}^i - 1 \quad s.t., \quad \min_{\forall ta \in T_i^{able}} \left\{ \left(l_{ta}^{L_{ta}^i} - l_{ta}^{L_{ta}^i - 1} \right) \times d_{ta}^i \right\} \tag{2}$$

The above equation checks the layer for cancellation. T_i^{able} indicates a set of tasks assigned higher layers than the compulsory minimum layer defined for fog node i. Namely, fog nodes should deal with at least the compulsory layer for their tasks, as shown in Equation (2). As some layers are canceled, the fog node regains its available capacity corresponding to the decreased loads. In contrast, more bandwidth will be required due to the larger number of intermediate results. The cancellation for the assigned layer is repeatedly carried out by Equation (2) until the fog node ensures computational capacity to handle the minimum layer for the task being attempted currently. If T_i^{able} is empty, i.e., assigned layers for all tasks on the fog node reaches the minimum layer, or a task has led to insufficient bandwidth on any fog node, the attempt to accommodate the new application to which this task belongs is considered a failure. Algorithm 1 indicates the pseudo code for the first procedure of the DLEFN, which defines the optimal layers for a new application.

Algorithm 1 The pseudo code for new application addition of DLEFN

/* input fog node ID i and new application n */
Input: i, n
Output: *accept*

init *accept* = **false**

foreach tn **in** T_n
  for $k = L_{tn}^{total}$ **to** L_i^{min} **step** -1
    if $r_{tn}^k \times d_{tn}^i / Q_{tn} \geq b_i^{able}$ **then**
      accept **= false**
      return *accept*

    else if $l_{tn}^k \times d_{tn}^i \leq c_i^{able}$ **then**
      $c_i^{able} = c_i^{able} - \left(l_{tn}^k \times d_{tn}^i\right)$
      $b_i^{able} = b_i^{able} - \left(r_{tn}^k \times d_{tn}^i\right)$
      $L_{tn}^i = k$
      accept **= true**
      break
    end if
  end for

  if *allow* is **false then**
    while T_i^{able} is not **empty**
      foreach ta **in** T_i^{able}
        $requiredOverhead = \left(l_{ta}^{L_{ta}^i} - l_{ta}^{L_{ta}^i - 1}\right) \times d_{ta}^i$
        $requiredBandwidth = \left(r_{ta}^{L_{ta}^i} - r_{ta}^{L_{ta}^i - 1}\right) \times d_{ta}^i / Q_{ta}$

        if ta **has smallest** *requiredOverhead* **among** T_i^{able} **and** $L_{ta}^i >= L_i^{min}$ **then**
          $L_{ta}^i = L_{ta}^i - 1$
          $c_i^{able} = c_i^{able} + requiredOverhead$
          $b_i^{able} = b_i^{able} - requiredBandwidth$
          break
        end if
      end foreach

      if $b_i^{able} < 0$ **then**
        accept **= false**
        return *accept*

      else if $l_{tn}^{L_i^{min}} \times d_{tn}^i \leq c_i^{able}$ **then**
        $c_i^{able} = c_i^{able} - \left(l_{tn}^{L_i^{min}} \times d_{tn}^i\right)$
        $b_i^{able} = b_i^{able} - \left(r_{tn}^{L_i^{min}} \times d_{tn}^i\right)$
        $L_{tn}^i = L_i^{min}$

        accept **= true**
        break
      end if
    end while
  end if
end foreach

return *accept*

In the case of a failure to accommodate a new DL application in first procedure, the other procedure of the DLEFN deals with giving a chance to the maximum number of DL applications. Commonly, the volumes of incoming data are related to the computational loads and the bandwidth required for transmission; larger volumes require greater resources. Accordingly, if an allowed application leads to a greater average incoming data volume than the new application for all fog nodes, the application is eliminated in the competition for DL service, and the new application is once again attempted by first procedure. Algorithm 2 indicates the pseudo code for application replacement. The average data volumes of all fog nodes for an allowed application a and a new application n are denoted by d_a^F and d_n^F, respectively.

Algorithm 2 The pseudo code for application replacement of DLEFN

```
/* input new application n */
Input: n
Output: replace

init replace = false

foreach a in A
  if d_a^F is largest and d_a^F > d_n^F then
    A.eliminate(a)
      foreach i in F
        replace = implement Algorithm1(i, n)
        if replace is false then
          break;
        end if
      end foreach
      break
  end if
end foreach

return replace
```

5. Performance Evaluation

5.1. Experimental Environment and Scenarios

Initially, we chose a DL framework called Caffe [21] (convolutional architecture for fast feature embedding), which has modularity and fast speed, for the experiment. Caffe was first developed by Yangqing Jia and, currently, hundreds of developers are involved in development of this open-source framework in github. The CNN (convolutional neural network) architecture in Caffe begins with an input layer followed by convolution (CONV) and pooling (POOL) layers, and terminating in fully connected (FC) layers as shown in Figure 7. The convolution layer computes dots between the input image and the entries of the filter that extend to the full depth to the image, and the pooling layer progressively reduces the spatial size of the representation. ReLU is most popular activation function, which transforms the weighted sum. The CONV, POOL and ReLU layers perform a role as learnable features extractor. On the other hands, the FC layers perform a role as a machine learning classifier.

After choosing Caffe for the simulation, we defined several custom models by revising the setting parameters of CNN models offered by Caffe offers and set up the solver parameters to optimize the models. After the definition of models and solver, all models were trained for the prepared data. Subsequently, we collected web images from Google, ImageNet, and Pixabay via a web-crawling method, and then executed custom models on the images to predict the outcomes. In this execution, we gathered operational information about required computational loads and data reduction ratios for each layer of all models. The information was utilized as reference in order to imitate the operation

of DL layers in the simulation. Figure 8 illustrates the procedures used for definition of DL custom models and gathering operational information of the defined models.

The performances of the proposed DLEFN algorithms were evaluated by simulators built based on the above operational information. The computing performance for each fog node was defined as 290–750Gflops, in accordance with the Tegra model of NVIDIA®. Allowed bandwidth for each fog node was restricted up to 10 Mbps, 100 Mbps, or 1 Gbps. The compulsory minimum layer was defined as 2 for all fog nodes, based on a pretested best value. An application consisted of one to four DL tasks that generated 5 MB to 150 MB per second and required a delay smaller than 200 ms. The proposed DLEFN was evaluated and compared with Li's offline and online scheme and methods supported by cloud only or by assigning two layers for all tasks. In this experiment, 500 applications requiring DL was used in random order. In addition, experiments with 30, 40, 50, 60, and 70 fog nodes were carried out to investigate the influence of the number of allowed DL applications in relation to number of fog nodes.

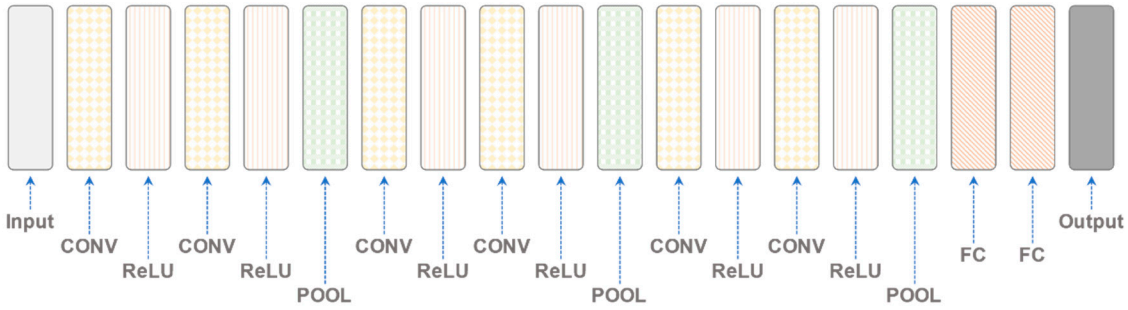


Figure 7. Example of convolutional neural network architecture.

(**a**)

(**b**)

Figure 8. Definition of DL custom models and gathering operational information of defined models. (**a**) A procedure for definition of DL custom models. (**b**) A procedure for gathering operational information of defined models.

5.2. Results and Discussion

Figure 9 shows the number of DL applications accepted in the methods mentioned in the previous section. The method supported by cloud only recorded the lowest number of allowed applications, due to insufficient bandwidth caused by the huge volumes of original data. Since this method processed original data without support by fog nodes, the comparison was unfair. Nevertheless, this result shows the necessity of fog computing in modern IoT environments that generate colossal volumes of data. The method of defining fixed layers for all tasks steadily allowed up to 160 applications. However, it could not accommodate any further applications. Despite being a simple algorithm that supports all tasks equally, this method allowed many more applications than processing the applications on the cloud only. In the offline method, all application tasks are arranged in ascending order of incoming average data size before implementing the algorithm. For that reason, the offline method steadily allowed up to around 250. The online algorithm utilizes historic data recorded by executing the offline algorithm in a real-time environment. Although the online algorithm recorded fewer allowed applications than the offline algorithm, the result was relatively higher than the methods supported by cloud only or by fixed layers. In contrast, the proposed DLEFN continuously accommodated applications despite the exhausted resources on all fog nodes, by first considering maximum resource utilization within fog nodes and comparing new applications and allowed applications to decide replacements for better efficiency. At the end, the DLEFN allowed highest number of applications requiring DL.

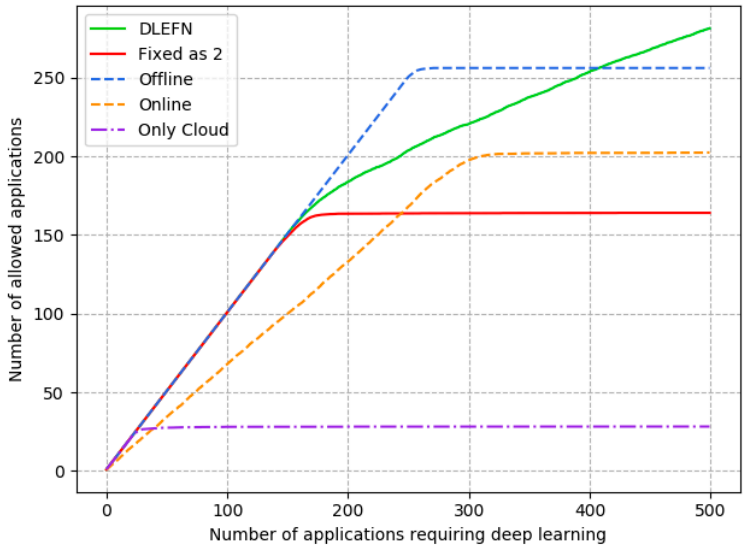
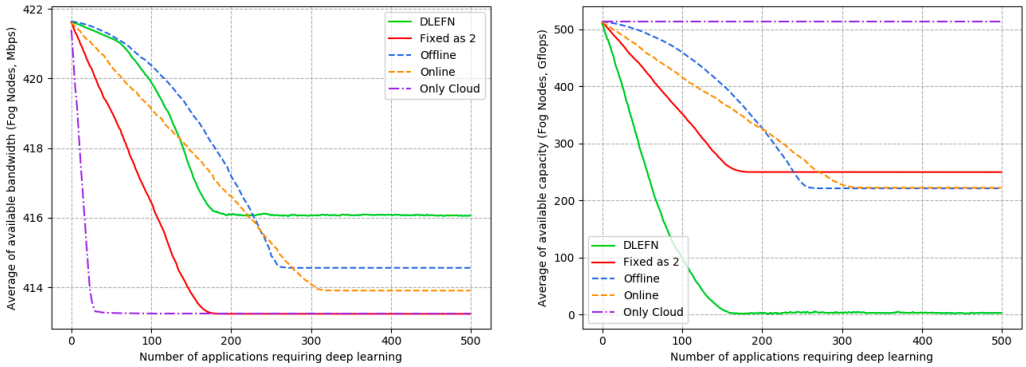


Figure 9. Number of allowed DL applications in fog computing (FC) environment.

Figure 10 shows the change of average available bandwidth and computational capacity of all fog nodes as each method decided whether to accommodate DL applications. As Figure 10a indicates, the final residual bandwidth on the network was significantly different for each algorithm. Nonetheless, the overall resources of all fog nodes were virtually exhausted by the proposed DLEFN, as shown in Figure 10b. In actuality, the proposed method was designed to utilize maximum resources at each fog node, although each node had different available resources. Consequently, fog nodes dealt with the optimal DL layer according to their available resources and tried to reduce the number of intermediate results. Therefore, the DLEFN gave a chance to more applications requiring DL service within given resources, as shown in Figures 9 and 10, since the proposed algorithm saved more available bandwidths than other methods even after allowing the largest number of applications. Accordingly, we can claim that the proposed method successfully minimized intermediate results being delivered to the cloud in an FC-based smart agriculture environment. As a result, decreased network congestion and computational loads on the cloud ensured better QoS.



(**a**) (**b**)

Figure 10. A change of average available bandwidth and computational capacity on fog nodes. (**a**) A change of average available bandwidth. (**b**) A change of average available computational capacity.

Figure 11 shows the average load caused by DL applications in the cloud after each method allowed all possible applications. Since more DL layers were handled at the fog nodes, computational loads per DL application were lowest in the proposed method. Figure 12 shows the influence of the number of allowed DL applications by number of fog nodes. Since an increase of fog nodes means an increase of available resources, the number of allowed applications was proportionally increased according to increasing number of fog nodes in all methods. The proposed method allowed DL service for the largest number of DL applications, regardless of the number of fog nodes.

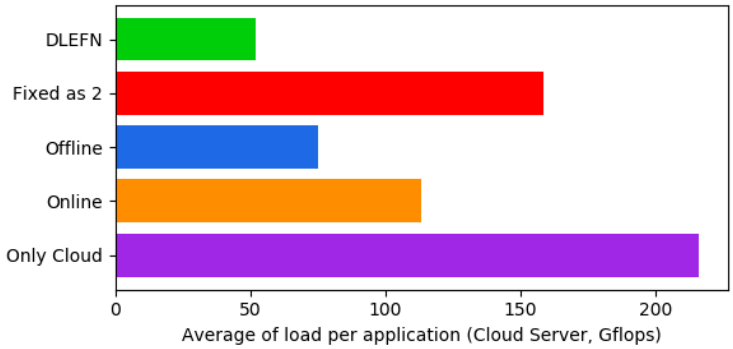


Figure 11. The average of load per application in the cloud server.

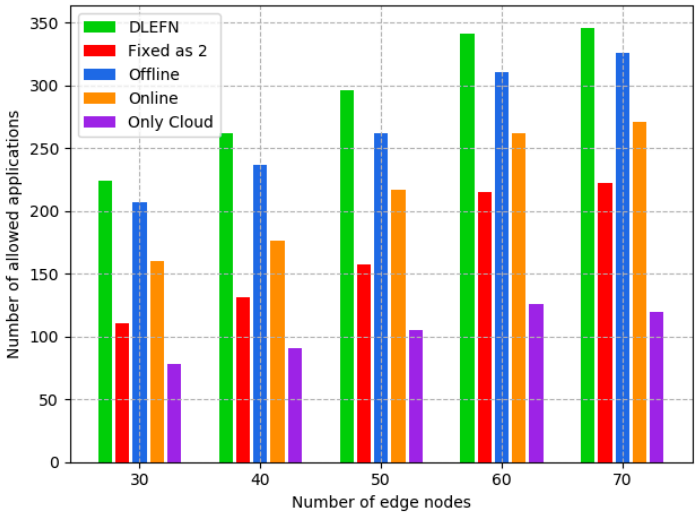


Figure 12. Number of allowed DL applications by number of fog nodes.

6. Conclusions

Colossal volumes of multimedia data gathered by IoT devices in modern smart agriculture are utilized to improve QoL of farmers. DL has come to the fore as a group of important techniques with which to pick out useful information from huge volumes of data. Nevertheless, supporting DL on IoT devices is constrained by their restricted resources and energy. Generally, cloud computing has been regarded as a universal approach to this issue. However, there is concern about service delay by network congestion caused by huge data volumes generated constantly by modern IoT infrastructures. Hence, we proposed a DLEFN algorithm that performs some DL tasks on fog nodes in an FC environment. The proposed method aimed to accommodate the maximum number of applications requiring DL service by using the maximum available resources of fog nodes, and to reduce network traffic and burden on the cloud. Accordingly, the DLEFN was designed to decide the optimal layers of DL model for application tasks for each fog node, in order to accommodate the maximum number of DL applications. The algorithm decided the appropriate layers of DL model for each fog node, taking into account its available bandwidth and computational capacity. The evaluated results were compared with existing methods in the same experimental environment and indicated considerable improvements regarding bandwidth efficiency, capacity efficiency, number of allowed DL applications, and cloud overhead. Considerable gains of performance proved the positive effect of individual decision-making about available resources of fog nodes. Therefore, as a solution among existing issues, the proposed algorithm excellently supported fog nodes with dissimilar resource capacities. Finally, we can clearly claim that the proposed DLEFN algorithm is an efficient scheme to graft DL onto FC-based smart agriculture environments to enable efficient service without service delays due to network congestion.

Author Contributions: Conceptualization, K.L., B.N.S. and K.H.; Data curation, K.L.; Formal analysis, K.L.; Investigation, K.L. and B.N.S.; Methodology, K.L.; Software, K.L.; Supervision, K.H; Writing—original draft, K.L.; Writing—review & editing, K.L, B.N.S and K.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by [National Research Foundation of Korea(NRF) grant funded by the Korea government (MSIT)] grant number [2019R1F1A1042721] and [BK21 Plus project (SW Human Resource Development Program for Supporting Smart Life) funded by the Ministry of Education, School of Computer Science and Engineering, Kyungpook National University, Korea] grant number [21A20131600005].

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Farooq, M.S.; Riaz, S.; Abid, A.; Abid, K.; Naeem, M.A. A Survey on the Role of IoT in Agriculture for the Implementation of Smart Farming. *IEEE Access* **2019**, *7*, 156237–156271. [CrossRef]
2. Ayaz, M.; Uddin, M.A.; Sharif, Z.; Mansour, A.; Aggoune, E.M. Internet-of-Things (IoT)-Based Smart Agriculture: Toward Making the Fields Talk. *IEEE Access* **2019**, *7*, 129551–129583. [CrossRef]
3. Elijah, O.; Rahman, T.A.; Orikumhi, I.; Leow, C.Y.; Hindia, N. An Overview of Internet of Things (IoT) and Data Analytics in Agriculture: Benefits and Challenges. *IEEE Internet Things J.* **2018**, *5*, 3758–3773. [CrossRef]
4. Zhang, P.; Zhao, Q.; Gao, J.; Li, W.; Lu, J. Urban Street Cleanliness Assessment Using Mobile Edge Computing and Deep Learning. *IEEE Access* **2019**, *7*, 63550–63563. [CrossRef]
5. Narvaez, F.Y.; Reina, G.; Torres-Torriti, M.; Kantor, G.; Cheein, F.A. A Survey of Ranging and Imaging Techniques for Precision Agriculture Phenotyping. *IEEE/ASME Trans. Mechatron.* **2017**, *22*, 2428–2439. [CrossRef]
6. Naseer, S.; Saleem, Y.; Khalid, S.; Bashir, M.K.; Han, J.; Iqbal, M.M.; Han, K. Enhanced Network Anomaly Detection Based on Deep Neural Networks. *IEEE Access* **2018**, *6*, 48231–48246. [CrossRef]
7. Yu, W.; Liang, F.; He, X.; Hatcher, W.G.; Lu, C.; Lin, J.; Yang, X. A Survey on the Edge Computing for the Internet of Things. *IEEE Access* **2018**, *6*, 6900–6919. [CrossRef]
8. Liu, J.; Chai, Y.; Xiang, Y.; Zhang, X.; Gou, S.; Liu, Y. Clean energy consumption of power systems towards smart agriculture: Roadmap, bottlenecks and technologies. *CSEE J. Power Energy Syst.* **2018**, *4*, 273–282. [CrossRef]

9. Shiruru, K. An introduction to artificial neural network. *Int. J. Adv. Res. Innov. Ideas Educ.* **2016**, *1*, 27–30.
10. Fadlullah, Z.M.; Tang, F.; Mao, B.; Kato, N.; Akashi, O.; Inoue, T.; Mizutani, K. State-of-the-Art Deep Learning: Evolving Machine Intelligence Toward Tomorrow's Intelligent Network Traffic Control Systems. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2432–2455. [CrossRef]
11. Zhou, X.; Li, S.; Tang, F.; Hu, S.; Lin, Z.; Zhang, L. DANoC: An Efficient Algorithm and Hardware Codesign of Deep Neural Networks on Chip. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 3176–3187. [CrossRef]
12. Silva, B.N.; Khan, M.; Han, K. Internet of Things: A Comprehensive Review of Enabling Technologies, Architecture, and Challenges. *IETE Tech. Rev.* **2017**, *2*, 205–220. [CrossRef]
13. Lane, N.D.; Georgiev, P.; Qendro, L. Deepear: Robust Smartphone Audio Sensing in Unconstrained Acoustic Environments Using Deep Learning. In Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing, Osaka, Japan, 7–11 September 2015; pp. 283–294.
14. Alsheikh, M.A.; Niyato, D.; Lin, S.; Tan, H.; Han, Z. Mobile Big Data Analytics Using Deep Learning and Apache Spark. *IEEE Netw.* **2016**, *30*, 22–29. [CrossRef]
15. Gupta, H.; Dastjerdi, A.V.; Ghosh, S.K.; Buyya, R. iFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *J. Softw. Pract. Exp.* **2017**, *47*, 1275–1296. [CrossRef]
16. Liu, C.; Cao, Y.; Luo, Y.; Chen, G.; Vokkarane, V.; Yunsheng, M.; Chen, S.; Hou, P. A New Deep Learning-Based Food Recognition System for Dietary Assessment on An Edge Computing Service Infrastructure. *IEEE Trans. Serv. Comput.* **2018**, *11*, 249–261. [CrossRef]
17. Lavassani, M.; Forsström, S.; Jennehag, U.; Zhang, T. Combining Fog Computing with Sensor Mote Machine Learning for Industrial IoT. *Sensors* **2018**, *18*, 1532. [CrossRef] [PubMed]
18. Li, H.; Ota, K.; Dong, M. Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing. *IEEE Netw.* **2018**, *32*, 96–101. [CrossRef]
19. Pan, J.; McElhannon, J. Future Edge Cloud and Edge Computing for Internet of Things Applications. *IEEE Internet Things J.* **2018**, *5*, 439–449. [CrossRef]
20. Wang, S.; Zhang, X.; Zhang, Y.; Wang, L.; Yang, J.; Wang, W. A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications. *IEEE Access* **2017**, *5*, 6757–6779. [CrossRef]
21. Komar, M.; Yakobchuk, P.; Golovko, V.; Dorosh, V.; Sachenko, A. Deep Neural Network for Image Recognition Based on the Caffe Framework. In Proceedings of the 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP), Lviv, Ukraine, 21–25 August 2018.