Distributed Artificial Intelligence enabled by oneM2M and Fog Networking

Kun Lun Cai and Fuchun Joseph Lin
Department of Computer Science
College of Computer Science
National Chiao Tung University
Hsinchu, Taiwan
{moon50102.cs05g, fjlin}@nctu.edu.tw

***Abstract*— Deep learning enabled by neural networks has been proven to be an effective Artificial Intelligence (AI) algorithm in sophisticated applications. The algorithm is normally divided into two phases: learning phase and inference phase. In this research, we assume the learning phase is already accomplished offline and focus on expediting the inference phase by replacing the centralized processing of Cloud with the distributed processing of Fog. In our approach, inference algorithms in AI are distributed to multiple layers of Fog networking, constructed from oneM2M Middle Nodes. We verify the performance improvement of our proposed distributed AI/Fog system by comparing it against a Cloud-centric system based on a use case of smart shopping mall.**

Keywords—Artificial Intelligence, Fog computing, IoT platform, oneM2M.

I. Introduction

Deep learning enabled by neural networks has been proven to be an effective Artificial Intelligence (AI) algorithm in sophisticated applications. Such an algorithm is normally divided into two phases: learning phase and inference phase. The wide deployment of IoT devices has enriched the input data required for the training phase. Once well-trained, during the inference phase the algorithms with the populated knowledge then can be applied for sophisticated applications such as speech and image recognition.

Such an AI system normally resides in the Cloud. This implies a large amount of sensing data need be transferred to the cloud for both learning and inference phases. This not only introduces significant delay in the response time of the AI system but also creates serious bandwidth shortage in the networks. Taking smart surveillance in a shopping mall as an example, there are many floors of large areas need to be monitored. As a result, thousands of smart cameras will be deployed in the shopping mall. Sending these data to a centralized server in the cloud will become infeasible [1]. Even if the problem of data transfer is solved, there will still be difficulty in processing image recognition in a reasonable time due to the large amount of data and the centralization of processing units.

In this research, we assume the learning phase is already accomplished offline and focus on expediting the inference phase by replacing the centralized processing of Cloud with the distributed processing of Fog. In our architecture, inference algorithms in AI are distributed to multiple layers of Fog networking, constructed from oneM2M Middle Nodes [2]. The term, 'Distributed Artificial Intelligence' is widely used in literature recently [3][4], but this term does not have an official definition. In our definition, we emphasize on the following key characteristics.

- Geographic distribution of computation resources
- Concurrent processing by independent nodes

The rest of the paper is organized as follows. In Section II, we will give a survey of related work and point out our unique contribution. In Section III, we will explain the high-level design of our proposed system and how it can be adopted to real-world use cases. Section IV then shows the implementation of our proposed system on a shopping mall use case. In Section V, the results of system evaluation will be presented. Finally, Section VI gives our conclusion and discusses future work.

II. Related Work

How to train a high accuracy model and how to use the model for inference over distributed architecture are two important research topics in AI. As a result, we categorize our survey into these two areas.

A. Learning over Distributed Architecture

In this research area, there are many papers [5][6] proposing different ways to coordinate heterogeneous devices to perform training and combine results. In particular, [7] and [8] present the methods for training machine learning models in a distributed system; they implement and test their solutions with real data in order to compare its accuracy against that of a centralized system. Although our system is focused on the inference phase, these papers provide good inspiration for our research.

B. Inference over Distributed Architecture

With the emergence of IoT, many large-scale applications such as smart city [9][10], smart farming [10], are introduced. Inside many of these IoT systems, AI algorithms are widely used. For example, [11] introduced hierarchical Fog Computing architecture for big data analysis in which a large-scale smart city use case is mapped into this architecture for performing big data analysis to identify hazardous events. Also in [12], neural networks are used for image processing to detect plant disease and fruit grading in smart farming.

As mentioned before, we focus on expediting the inference phase. To achieve our objective, we explore the approach of inference over distributed architecture. Our approach is characterized by the following unique features: (1) we replace the centralized processing of Cloud with the distributed processing of Fog, (2) we use standardized IoT platform, oneM2M, as the communications middleware between different layers of Fog nodes, (3) we adopt DNN (Deep Neural Network) as our inference engine and (4) we solve the problem of transferring large data between different layers of Fog architecture.

III. Proposed System Architecture

In this section, we explain the high-level design of our system including its functions and workflows. The functional architecture of the system is shown in Figure 1 where all the components required to construct an AI inference engine over distributed architecture are illustrated. This is a Fog distributed system of hierarchical architecture onto which a large-scale application can be decomposed and deployed. Starting from the bottom is Sensor Layer that collects the sensing data. Then each of lower layers in turn would produce data for the next higher layer until it reaches Cloud Layer.

A. Layer-by-layer Explanation

Our system consists of three layers: Sensor Layer, Fog Layer, Cloud Layer and Actuator Layer.

1) Sensor Layer

This is where the sensing data come from. Various types of sensors can be deployed in this layer. The system supports not only the sensors that generate simple raw data but also the ones that generate large data like videos and images. The task in this layer is just collecting and sending the data. Any further processing will be performed in the higher layers.

2) Fog Layer

This is where the 'edge intelligence' resides. Fog Layer receives the input data from Sensor Layer and performs 'intelligent' inference to produce useful insights. The tasks of the Fog nodes in this layer may include data preprocessing, feature extraction, knowledge inference or other AI algorithms. As a result, the Fog nodes deployed at this layer should be general-purpose and capable of performing any tasks mentioned above.

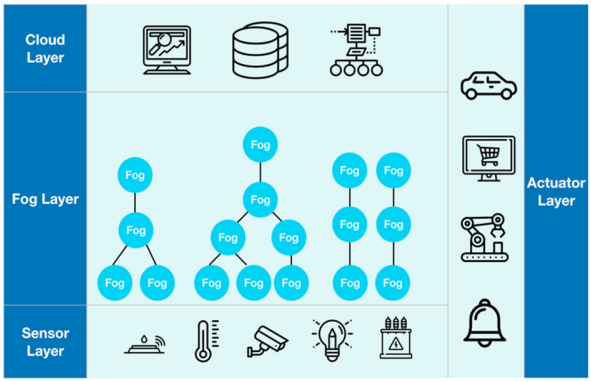

Figure 1. Proposed System Architecture

All these Fog nodes are architected in a hierarchical fashion as illustrated in Figure 1. The standardized IoT platform, oneM2M [2], is used as the communications middleware between the successive levels of Fog nodes. They may consist of several levels depending on the requirements of use cases. For example, if the data in the use case require three different steps of processing, we can use three-level architecture. With Fog Layer, we can filter out lots of useless data and save significant network bandwidth by sending only a small amount of useful data to Cloud Layer.

3) Cloud Layer

Depending on the use case, Cloud Layer would execute different AI algorithms for action decision. For instance, if the disease detection in smart farming is positive, the application in the cloud should decide how to react to this situation. It can either dispatch the cure or just send an alert to the farmer.

4) Actuator Layer

The actuators in this layer receive the commands or data from Cloud Layer and carry out the actions. Note that the nodes at this layer normally do not perform any analysis. Some examples are shown in Figure 1 including a monitor to display the advertisement, a vehicle or robot to execute the command, or an alarm that simply plays the siren.

In Section IV, we will introduce a use case implementation of smart shopping mall to demonstrate the flexibility and generality of our design. It illustrates how a real-world use case can be mapped to our proposed system architecture.

B. Inter-Layer Communications

To add a hierarchical system of Fog nodes between IoT devices and the Cloud, a middleware is required to support connectivity and communications among these nodes. We propose to use the standardized IoT platform, oneM2M [2], as the middleware. There are several open source implementations of oneM2M and we have chosen OM2M [13], developed by LAAS-CNRS, in our initial trial. Though OM2M is not designed for efficiently handling big data such as images or videos, we alleviate this problem by designing an enhanced communication mechanism over OM2M.

In our design, each Fog node is constructed as an oneM2M Middle Node – Common Service Entity (MN-CSE) as shown in Figure 3. Then the AI inference engine is deployed as an Application Entity (named AI-AE) on top of MN-CSE for distributed task allocation. For example, the first level of Fog nodes can be used for data preprocessing and feature extraction from the raw data collected at the sensor layer (marked as "A"). The second level of Fog nodes then takes the input from the first level of Fog nodes (marked as "B") and performs further inference based on DNN models already trained. Then, the inference result of distributed Fog computing will be sent to the Cloud (marked as "C"). Finally, the last of inference algorithms will be carried out by the Cloud in order to generate the decision commands to the actuator layer (marked as "D").

In oneM2M, each MN-CSE can maintain its own resource-based information model called Resource Tree. Figure 4 shows an example of such resource trees maintained in MN-CSE. Each resource has its own Resource Type which represents the

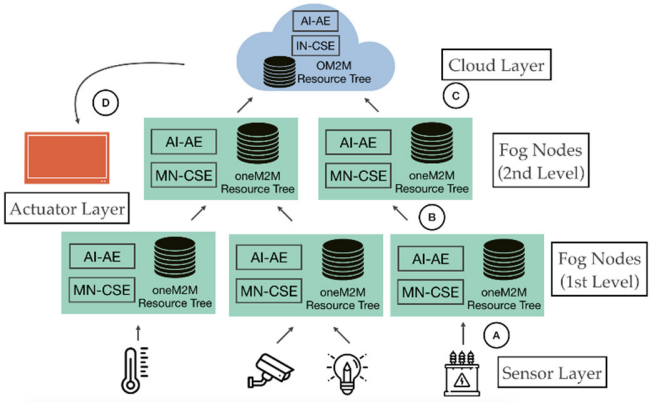

Figure 3. Detailed System Design

semantics of the resource. Whenever information is to be saved in oneM2M, it will be treated as "resource" and maintained in the resource tree. To manipulate resources, oneM2M provides Restful API to perform Create, Read, Update, Delete, Subscribe and Notify.

Here we explain some of the most important Resource Type shown in Figure 4.

1) <CSE-ID>

This is the ID of the MN-CSE or IN-CSE which the current MN-CSE registers to. In oneM2M, it is necessary to perform registration during the initialization phase of an MN-CSE.

2) <AE>

Basically, AE is the resource for the application that is managed by the current MN-CSE. For example, in Smart Home, there are many different services running in a house such as air conditioner control, smart light control. These services will register to MN-CSE as an AE.

3) <Container>

It is the resource maintaining multiple instances of real data. However, it can have more information than the raw data from sensors. For example, to provide automatic air conditioner control at Smart Home we not only keep track of temperature and humidity from sensors but also people count and other useful information. These data will be maintained in different Containers.

4) <contentInstance>

This is the resource that stores a single instance of real data like "30 degree Celsius". It is created as a sub-resource of Container.

5) <Subscription>

The <Subscription> resource is the key mechanism in oneM2M that enables inter-node communications in our system. Creating this resource as a child under a parent resource means to subscribe to any update in the parent source. Once a resource is subscribed, oneM2M will monitor the status of the resource and notify the subscribed AE whenever there is any new update to the resource. We make heavy use of this oneM2M subscription and notification mechanism for inter-

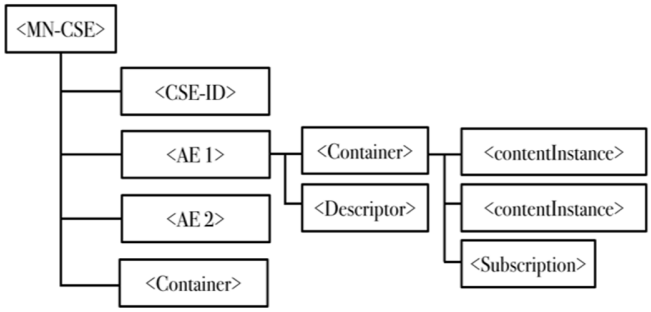

Figure 4. Example of oneM2M Resource Tree

Fog nodes communications [14]. With this mechanism, data transmission among different levels of Fog nodes and between Fog nodes and the Cloud can be easily achieved.

Although our adopted oneM2M open source implementation, OM2M, can manage the resources in small scale without problems, it lacks the capability of dealing with large data files such as videos. We thus design an enhanced communication mechanism based on Secure Copy Protocol (SCP) for data transfer. In this design, the information stored in oneM2M container/contentInstance has become the path of the large data file. A subscriber will receive the notification that contains only path information, but with this path information a subscriber will be able to set up an SCP connection and retrieve the actual transfer data.

IV. Use Case Implementation

We have implemented our system architecture described in Section III by applying it for a smart shopping mall use case. In this use case, the system objective is to increase the effectiveness of advertisement broadcasting in a shopping mall by estimating the distribution of age/gender for the incoming customers in each area and each floor of the mall. Here we assume a shopping mall with multiple floors and each floor with many stores; cameras are deployed in front of every store to capture customers' videos. Also, electronic billboards are placed on each floor of the mall to display advertisements. These advertisements will change dynamically according to the age/gender distribution of the customers. We assume the association between advertisement and age/gender has been established before system operations.

For this use case, two levels of Fog nodes in a hierarchical structure are constructed between Sensor Layer and Cloud Layer. The first level of Fog nodes is used to detect and retrieve the faces of customers from the video files collected by the cameras. The second level of Fog nodes then take the face inputs from the first level of Fog nodes and perform analysis based on Deep Neural Network (DNN) models to decide ages and genders of customers. The deployment of the Fog nodes can be quite flexible. For example, we can deploy the Fog nodes based on the layout of shopping mall. If the shopping mall has multiple floors and each floor has multiple areas, one first-level Fog node can be deployed in each area and one second-level Fog node can be deployed on each floor. Also, depending on the business nature of the area, more Fog nodes can be deployed

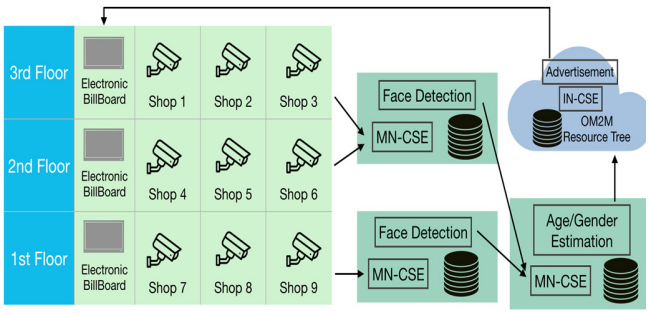

Figure 6. High-level Architecture of System Implementation

there to deal with unusual situations such as in the area of food court where it often attracts more crowd.

After finding out the ages/genders of the customers, this information will be sent to Cloud Layer for further analysis in order to decide the best advertisement for Actuator Layer. The high-level system architecture is shown in Figure 2. To explain the data flow within the system more clearly, we will go through the lifecycle of data (video) in the following subsections.

A. Video to Face

In our experiment, the cameras are simulated by webcams on the laptops. As such, the captured video is saved in a file on the laptop directly. The paths of those files will be stored by the camera application in the resource tree of the MN-CSE on the laptop that captured the video. The first level of Fog nodes will subscribe to the containers designed to keep the paths of these video files. Thus they will be notified about the availability of video files (in terms of path names) on the resource tree. Whenever this happens, the first level of Fog nodes will follow these paths to retrieve the video files via SCP, perform face detection and extract face information. In our implementation, we utilize "OpenCV"[15] to detect the faces from videos and 'pickle'[16] to convert the face data for storage and transmission. By extracting only the relevant data from these large video files, the first level of Fog nodes would filter out useless data in the initial video files and reduce the amount of data to be transferred to the next level of Fog nodes. After the faces in each video frame are found, they will be cut out and saved into a separate file. The paths of these face files will again be saved in the local containers that have been subscribed by the Fog nodes in the second level.

B. Face to Age/Gender

The second level of Fog nodes will be notified about the availability of face data stored in the resource tree of the first level of Fog Node. They will retrieve the paths of these face files, follow these paths to retrieve the files and perform age/gender detection by analyzing the data in these files. Here the Fog nodes will use a pre-trained DNN model to estimate the ages and genders of the given faces. The neural network we use is WideResNet trained by [17]. The DNN libraries we use is 'Keras'[18] based on 'TensorFlow'[19] and 'Theano'[20]. The environment is installed, configured and tested on Ubuntu 16.04 and macOS 10.13. After the processing, all the ages and genders of the faces will be saved on the local resource tree that is subscribed by Cloud Layer.

C. Age/Gender to Advertisement

The age and gender data will be used by the Cloud. When this data is ready, the Cloud will receive the notification with age and gender data; then the Advertisement AE will perform analysis to find out the majority of the customers in terms of age and gender. In our experiment, we divide the ages from 10 to 70 into 12 groups for each gender and map these to 24 advertisements with one for each group. With this mechanism, the most effective advertisement that target at the specific age and gender can be chosen for display to maximize the efficiency and accuracy of the advertisement.

D. Advertisement Display

In our implementation, we design an electronic billboard application to complete our use case. This electronic billboard will receive the analysis result from the Cloud via a TCP connection and display the best suited advertisement to encourage the customer to buy. In a shopping mall, the electronic billboards can be placed in strategic locations such as at the entrance or in the customer help desk area to display special promotion advertisement. In addition, to achieve effective promotion, we have to constantly track the change of customer composition and update the advertisement dynamically as needed.

V. System Evaluation

To verify the efficiency of our proposed system vs. that of the traditional cloud-centric system, we define three evaluation metrics: amount of data transferred, length of execution time for face/age/gender determination, and length of end-to-end response time. These three metrics are chosen due to the following considerations: First, the amount of data transferred is the major problem we want to solve to prevent network congestion as mentioned from the very beginning. Second, one of the biggest differences between the centralized and the distributed systems is how a task is separated into several subtasks. As a result, the execution time of these tasks plays an important factor as we compare two systems. Third, in our proposed system, we add Fog Layer between end-devices and Cloud Layer. With this design, the generated data can be processed immediately without traveling far to the cloud. Consequently, the length of end-to-end response time is also used as one of the metrics for comparison.

The computer environment used for centralized and distributed system testing is shown in Table 1. To ease our testing procedure, we have sampled six short videos of various people walking flows with each lasting 5-10 seconds as our testing data. Each video has different characteristics that can simulate diverse environments in a shopping mall. We separate these videos into three categories which are corridor, shopping area, and food court. The sampled videos contain various mixtures of people with different ages and genders. As a result, we can simulate various distributions of age and gender in different areas of a shopping mall and test the capability of

Table 2. Tested Environment

	Operating system	CPU	RAM	GPU
Centralized System	macOS 10.31	Intel(R) Core(TM) i5-6360U CPU @ 2.00GHz	8192 MB	Intel Iris Graphics 540 1536 MB
Distributed System	Ubuntu 16.04	Intel(R) Core(TM) i5-6360U CPU @ 2.00GHz	4096 MB	Disabled

displaying the advertisements dynamically depending on the age/gender distribution.

A. Amount of Data Transfer

In our distributed system there are two additional levels of Fog nodes between end devices and the cloud. The first level of Fog nodes will perform face extraction while the second level of Fog nodes will analyze the faces using a pre-trained DNN model to estimate their ages and genders. After being processed by these two levels of Fog nodes, the original large video file would eventually be reduced to a short string (e.g. <32, Male>, <45, Female>). Figure 5 illustrates how much data transfer is reduced after being processed by each level of Fog Nodes. The size of video file, 5500 KBs, is the amount of data to be transferred from Sensor Layer to Cloud Layer in a cloud-centric system. However, in our proposed system, the results show that the first level of Fog nodes is able to reduce about 50% of total data amount while the second level of Fog nodes can reduce up to 99.9997% of total data amount. This truly demonstrates a great saving on network resources by a Fog system.

B. Execution Time for Face/Age/Gender Determination

In a Fog distributed system the program execution flow is separated into two phases (face detection and age/gender estimation). Hence, the execution time has to be calculated for each phase individually; then two results would be summed up to derive the total execution time.

Figure 7 shows a comparison of the execution time between the centralized system and the distributed system for one video file processing. As we can see from the result, the performance

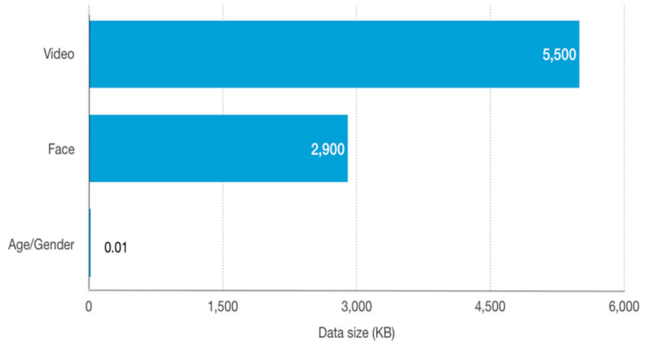

Figure 8. Data Size Transferred between each Layer

of the centralized system is slightly better than that of the distributed system due to the better computational resources of the centralized system. Nevertheless, the difference is only three percent, indicating that the distributed Fog system is a viable choice even for the extreme case of processing one video file.

In the case of processing multiple video streams, the Fog architecture will exhibit much better execution time by utilizing parallel computation enabled by multiple Fog nodes at the same level.

C. End-to-End Response Time

One of the major differences between Cloud and Fog Computing is where the sensor data will be processed. In the Cloud architecture, the data need to be sent a long way to Cloud for processing. In Fog Computing, Fog nodes are deployed much closer to Sensor Layer, so the network condition is much more stable compared to the centralized architecture. During the preprocessing executed on Fog nodes, the input data will become smaller when they are sent to the higher layer of Fog nodes. As a result, the large data on lower layer will be sent through the stable network while only a small amount of data will be transferred from Fog Layer to Cloud Layer as we showed in Section V-(A).

In our testbed, we use Wi-Fi to connect the devices. To simulate the difference between the centralized system and the distributed system, we weaken Wi-Fi signals to slow down the transmission speed to simulate the long haul communication between Fog and Cloud but use normal signal strength to simulate the short distance communication among Fog nodes and between devices and Fog nodes. In our experiment, we are able to simulate the normal speed at 5.3MBps and the lower speed at 0.67MBps. With the results shown in Section V-(A), we can estimate the end-to-end response time of centralized and distributed system. In the centralized system, video files are directly sent from Sensor Layer to Cloud Layer, so the estimated transmission time will be the size of video file divided by transmission speed. The result is 8.209 seconds. In the distributed system, the video files need to go through Sensor

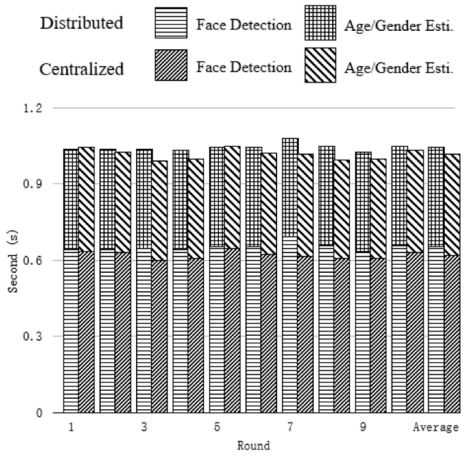

Figure 7. Execution Time of Centralized vs. Distributed System

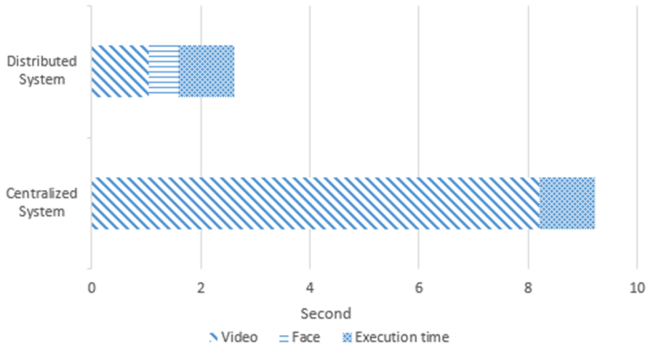

Figure 10. End-to-end response time

Layer, two levels of Fog nodes, and Cloud Layer. However, the size of age/gender data is less than 0.01 KB, so we can ignore the data transmission time between second level of Fog nodes and Cloud Layer. Consequently, total data transmission time in our implementation will become the sum of the time to transmit video and face files. The final result will be 1.585 seconds which is 80.67% less than centralized system. This experiment demonstrates the benefits of data preprocessing and close proximity of fog networking.

Combining the results of execution time and end-to-end data transmission time, we can estimate the total response time which is the time required to make the input (video file) become the output (age/gender). The result is shown in Figure 9. In our testbed, the total response time of the centralized system is 9.219 seconds while that of the distributed system is 2.615 seconds which is 71.63% reduced. We thus believe the distributed system is a viable solution in implementing large scale use cases.

VI.Conclusion and Future work

In this paper, we introduced a distributed AI system enabled by oneM2M and Fog Networking. We discussed the construction of a hierarchical structure of Fog nodes between the Cloud and end devices by utilizing oneM2M as the communication middleware. Then we showed how AI inference algorithms can be distributed among Fog nodes to enable efficient IoT applications. We designed and tested our system using a use case of "smart shopping mall" where the ages and genders of visiting customers will be collected and analyzed in order to deliver the most effective advertisements. Our evaluation showed that this Fog-based AI system can solve several common problems such as lack of network and compute resources. In addition, the use of IoT platform, oneM2M, greatly enhanced the capability of connecting multiple level of Fogs and Cloud into a coherent system in a large-scale use case [21].

We believe our system design is general and flexible enough that it can be applied to many other use case scenarios. In the future, we plan to extend our proposed system to support the capability of distributed learning [22] and combine it with the capability of distributed inference discussed in this paper. Another potential improvement is to run our experiments with real-time video streaming than pre-stored video clips. Also, we believe it is worthwhile to explore the separation of neural networks inside our proposed system such that different parts of the neural networks can be executed on multiple Fog nodes in parallel in order to further increase execution speed.

Acknowledgement

The project reported in this paper is sponsored by Institute for Information Industry in Taiwan under the 2017 Initiative on *Docker-based OpenFog IoT Technologies* and the 2018 Initiative on *Gateway Site Fog Computing*.

References

[1] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," IEEE Internet of Things Journal, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[2] oneM2M official website : http://www.onem2m.org

[3] Definition of Distributed Artificial Intelligence: https://www.techopedia.com/definition/6720/distributed-artificial-intelligence-dai

[4] Definition of Distributed Artificial Intelligence: https://en.wikipedia.org/wiki/Distributed_artificial_intelligence

[5] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. In Proceedings of LearningSys, 2015.

[6] Provost, F.J., & Hennessey, D. (1996). Scaling up: Distributed machine learning with cooperation. Proceedings AAAI-96.

[7] Surat Teerapittayanon, Bradley McDanel, HT Kung, "Distributed Deep Neural Networks over the Cloud, the Edge and End Devices", Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on pages 328-339.

[8] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, Kevin Chan, "When Edge Meets Learning: Adaptive Control for Resource-Constrained Distributed Machine Learning", INFOCOM 2018.

[9] K. Su, J. Li, and H. Fu, "Smart city and the applications", International Conference on Electronics, Communications and Control (ICECC), pages 1028–1031, 2011.

[10] P. Tripicchio, M. Satler, G. Dabisias, E. Ruffaldi, C. Avizzano, "Towards smart farming and sustainable agriculture with drones", 2015 International Conference on, pp. 140-143, July 2015.

[11] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, Q. Yang, "A hierarchical distributed fog computing architecture for big data analysis in smart cities", ASE BD&SI '15: Proceedings of the ASE BigData & Social Informatics, pp. 28:1-28:6, 2015.

[12] Monica Jhuria, Ashwani Kumar, Rushikesh Borse, "Image Processing For Smart Farming: Detection Of Disease And Fruit Grading", Proceedings of the 2013 IEEE Second International Conference on Image Information Processing.

[13] OM2M official website : http://www.eclipse.org/om2m/

[14] oneM2M Technical Specification TS-0001 : Subscription and Notification, http://www.onem2m.org/images/files/deliverables/TS-0001-oneM2M-Functional-Architecture-V-2014-08.pdf

[15] OpenCV: https://opencv.org/opencv-3-0.html

[16] Pickle : https://docs.python.org/3/library/pickle.html

[17] Age gender estimator : https://github.com/yu4u/age-gender-estimation

[18] Keras : https://keras.io

[19] TensorFlow : https://www.tensorflow.org

[20] Theano : http://deeplearning.net/software/theano

[21] K. Datta, C. Bonnet, "A lightweight framework for efficient m2m device management in onem2m architecture", Recent Advances in Internet of Things (RIoT) 2015 International Conference on, pp. 1-6, April 2015.

[22] https://en.wikipedia.org/wiki/Online_machine_learning : Online machine learning