Intelligent Automation & Soft Computing
DOI:10.32604/iasc.2021.019329
Article

Tech Science Press

Performance Comparison of PoseNet Models on an AIoT Edge Device

Min-Jun Kim[1], Seng-Phil Hong[2], Mingoo Kang[1] and Jeongwook Seo[1,*]

[1]Department of IT Transmedia Contents, Hanshin University, Osan-si, 18101, Korea
[2]Hancom With Inc., Seongnam-si, 13493, Korea
[*]Corresponding Author: Jeongwook Seo. Email: jwseo@hs.ac.kr
Received: 09 April 2021; Accepted: 10 May 2021

Abstract: In this paper, we present an oneM2M-compliant system including an artificial intelligence of things (AIoT) edge device whose principal function is to estimate human poses by using two PoseNet models built on MobileNet v1 and ResNet-50 backbone architectures. Although MobileNet v1 is generally known to be much faster but less accurate than ResNet50, it is necessary to analyze the performances of whole PoseNet models carefully and select one of them suitable for the AIoT edge device. For this reason, we first investigate the computational complexity of the models about their neural network layers and parameters and then compare their performances in terms of inference time, frame rate and pose score. We found that the model with MobileNet v1 could estimate human poses very fast without significant pose score degradation when compared with the model with ResNet-50 through some experimental results. In addition, the model with MobileNet v1 could smoothly handle video clips with 480 × 360 and 640 × 480 resolutions as inputs except 1280 × 720 resolution. Finally, we implemented the overall oneM2M-compliant system with an AIoT edge device containing the model with MobileNet v1 and a floating population monitoring server, which is applicable to commercial area analysis in a smart city and we verified its operation and feasibility by periodically displaying all the data of temperature, humidity, illuminance, and floating population estimation from the AIoT edge device on a map in a web browser.

Keywords: Deep learning; edge computing; Internet of Things; MobileNet v1; PoseNet; ResNet-50

1 Introduction

Artificial intelligence of things (AIoT) is a new technology combining artificial intelligence (AI) with the Internet of Things (IoT) [1,2]. It enables the IoT devices to analyze their sensing data, make decisions and act on the decisions without human involvement. In addition, edge computing is known as a distributed computing technology bringing computation to the edge of an IoT network, where local IoT devices can process time-sensitive sensing data as close to its source as possible [3]. Its primary benefit is that it allows sensing data to be processed right on the spot, eliminates latency and responds instantaneously. It is able to address any concerns with regard to response time requirement, bandwidth cost saving, battery life constraint, data safety and privacy [4,5]. Moreover, the fusion of AIoT and edge computing can create a new AIoT edge device and this combination of edge computing and AI in an IoT device has great potential to be applicable in various fields [6–9].

Deep learning as a subfield of AI is usually considered in order to make an AIoT edge device intelligent [10]. Here we want to add a deep learning model for pose estimation to an AIoT edge device and the pose estimation model is a key component in enabling the AIoT edge device to have an understanding of people in images and videos. There are two approaches referred to as top-down and bottom-up for pose estimation [11]. Top-down approaches employ a person detector and then perform single person pose estimation for each detection. But they have no way to recovery if the person detector fails and their runtime highly depends on the number of people in the image for each person detection. In contrast, bottom-up approaches offer robustness to the recovery problem and may decouple runtime complexity from the number of people in the image. Traditional bottom-up approaches perform inference over a combination of local observations on body parts and the spatial dependencies between them. Convolutional neural networks (CNNs) have been widely used to obtain reliable local observations of body parts. PoseNet is one of CNN models for estimating human poses in which human poses are made from a group of 17 different and predefined body parts called keypoints [12,13] and it has two variants of backbone architectures known as MobileNet v1 and ResNet50. Generally, MobileNet v1 is much faster but less accurate when compared with ResNet50 but we need to analyze the performances of two PoseNet models with MobileNet v1 and ResNet50 to know which one is more suitable for AIoT edge devices with different hardware resources and applications.

Therefore, this paper presents an oneM2M-compliant system model with an AIoT edge device and a floating population monitoring server for commercial area analysis and store operation services, select the PoseNet model proper to the AIoT edge device through performance comparisons, and additionally shows some experimental results. The rest of the paper is organized as follows. In Section 2, a system model and its PoseNet models are presented. Also, oneM2M-compliant IoT entities and web monitoring application are briefly described. In Section 3, we compare the PoseNet models according to some performance metrics. In Section 4, the concluding remarks are given.

2 AIoT Edge Device and Its PoseNet Models

2.1 System Model with an AIoT Edge Device

In Fig. 1, an oneM2M-compliant system consisting of an AIoT edge device and a floating population monitoring server is illustrated.

The AIoT edge device receives temperature and humidity data from a DHT11 sensor, illuminance data from a BH1750 sensor and video data from a C270 high definition (HD) webcam. It estimates human poses in video data with PoseNet models that are based on MobileNet v1 and ResNet-50 architectures and then estimates floating population by counting estimated human poses. Finally, an IoT client application in the AIoT edge device sends all the data such as temperature, humidity, illuminance and floating population estimation data to an IoT server middleware in the floating population monitoring server. The IoT server middleware stores the data coming from the IoT client application and forwards them to a web monitoring application. The web monitoring application displays the coordinates of the AIoT edge device and the data on a map in a web browser.

2.2 Comparison of PoseNet Models

In terms of their neural network layers and parameters, we compare computational complexity of two PoseNet models based on MobileNet v1 and ResNet-50 architectures in the AIoT edge device. The PoseNet is a deep learning model to estimate human poses by detecting 17 keypoints which include

5 facial points such as two eyes, two ears and one nose and 12 joints of the body such as shoulders, elbows, wrists, hips, knees and ankles [14–16]. Here two PoseNet models are considered where the first model is built on a backbone network of MobileNet v1 and the second is built on a backbone network of ResNet-50 [17,18]. The MobileNet v1 or ResNet-50 block takes an input image with red, green and blue channels and outputs feature maps. From feature maps, the Keypoint Heatmaps block predicts a heatmap $p_k(x_i)$ with a binary classification task which is represented as

$$p_k(x_i) = 1 \;\text{ if }\; x_i \in \mathcal{D}_R\left(y_{j,k}\right) \;\text{ otherwise }\; p_k(x_i) = 0 \tag{1}$$

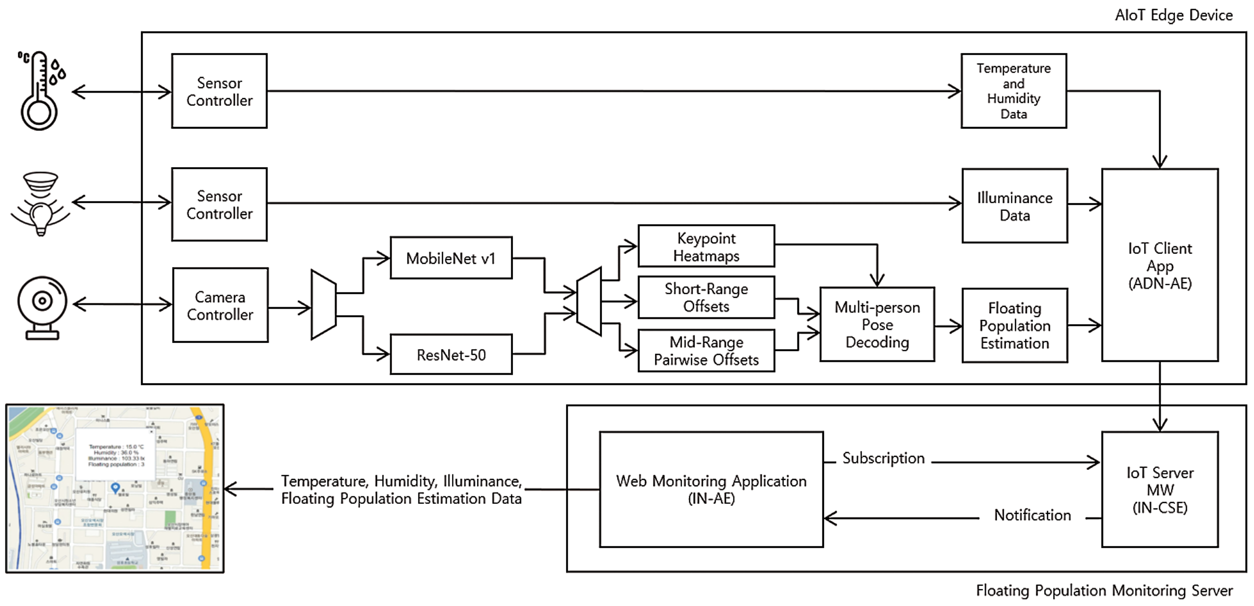

AIoT Edge Device
Sensor Controller
Temperature and Humidity Data
Sensor Controller
Illuminance Data
Camera Controller
MobileNet v1
ResNet-50
Keypoint Heatmaps
Short-Range Offsets
Mid-Range Pairwise Offsets
Multi-person Pose Decoding
Floating Population Estimation
IoT Client App (ADN-AE)
Temperature, Humidity, Illuminance, Floating Population Estimation Data
Web Monitoring Application (IN-AE)
Subscription
Notification
IoT Server MW (IN-CSE)
Floating Population Monitoring Server

Figure 1: oneM2M-compliant system with an AIoT edge device and a floating population monitoring server applicable to commercial area analysis in a smart city

Assuming that N is the number of pixels in the image, M is the number of person instances in the image, and K is the number of keypoint types in a person instance, x_i denotes the 2-D position for $i = 1, \ldots, N$, $y_{j,k}$ denotes the 2-D position of the k-th keypoint of the j-th person instance for $k = 1, \ldots, K$ and $j = 1, \ldots, M$, and $\mathcal{D}_R\left(y_{j,k}\right) = \left\{x : \| x - y_{j,k} \| \le R\right\}$ denotes a disk of radius $R = 32$ of centered on $y_{j,k}$. During training, the heatmap loss is computed as the average logistic loss along image positions. To improve the keypoint localization accuracy, the Short-Range Offsets block predicts short-range offset vectors given as

$$S_k(x) = y_{j,k} - x \tag{2}$$

which point from the image position x within the keypoint disks to the k-th keypoint of the closest person instance j. During training, the short-range offset prediction errors are penalized with the L_1 loss and the errors only at the positions $x \in \mathcal{D}_R\left(y_{j,k}\right)$ in the keypoint disks are averaged and back-propagated. To connect pairs of keypoints belonging to each person instance, the Mid-Range Offsets block predicts a separate pairwise mid-range 2-D offset vectors $M_{k,l}(x) = \left(y_{j,l} - x\right)\left[x \in \mathcal{D}_R\left(y_{j,k}\right)\right]$ which are $2(K-1)$ target regression vectors for each directed edge connecting pairs (k, l) of keypoints adjacent to each other in a tree-structured kinematic graph of the person. During training, the average L_1 loss of the regression prediction errors is computed and back-propagated through the network. To address accurate regressions, refining the mid-range pairwise offsets is repeated twice by using the more accurate short-range offsets, which is shown as

$$M_{k,l}(x) \leftarrow x' + S_l(x'), \quad \text{where } x' = M_{k,l}(x) \tag{3}$$

To sample $S_l(x')$, bilinear interpolation is employed. the errors through Eq. (3) are back-propagated along both the mid-range and short-range input offset branches. The Multi-person Pose Decoding block aggregates the heatmap and short-range offsets through Hough voting into 2-D Hough score maps $h_k(x)$ given as

$$h_k(x) = \frac{1}{\pi R^2} \sum\nolimits_{i=1:N} p_k(x_i) B(x_i + S_k(x_i) - x) \tag{4}$$

where $B(\cdot)$ is the bilinear interpolation kernel. The position x_i and keypoint type k of all local maxima in the Hough score maps $h_k(x)$ are inserted in a priority queue and then they are popped out of the queue in descending score order. At each iteration, if the current candidate seed x_i of the keypoint type k is within a disk $\mathcal{D}_r\left(y_{j',k}\right)$ of previously detected person instances j', it will be rejected by using a non-maximum suppression radius of $r = 20$ pixels. Otherwise, a new person instance j with the keypoint type k is started at position $y_{j,k} = x_i$ as a seed and the pairs (k, l) of adjacent keypoints are greedily connected along the edges of the kinematic person graph, which is represented as

$$y_{j,l} = y_{j,k} + M_{k,l}\left(y_{j,k}\right) \tag{5}$$

Fig. 2 shows the PoseNet models with MobileNet v1 and ResNet-50 backbone architectures in order to compare their computational complexity for a 640 × 480 video stream.

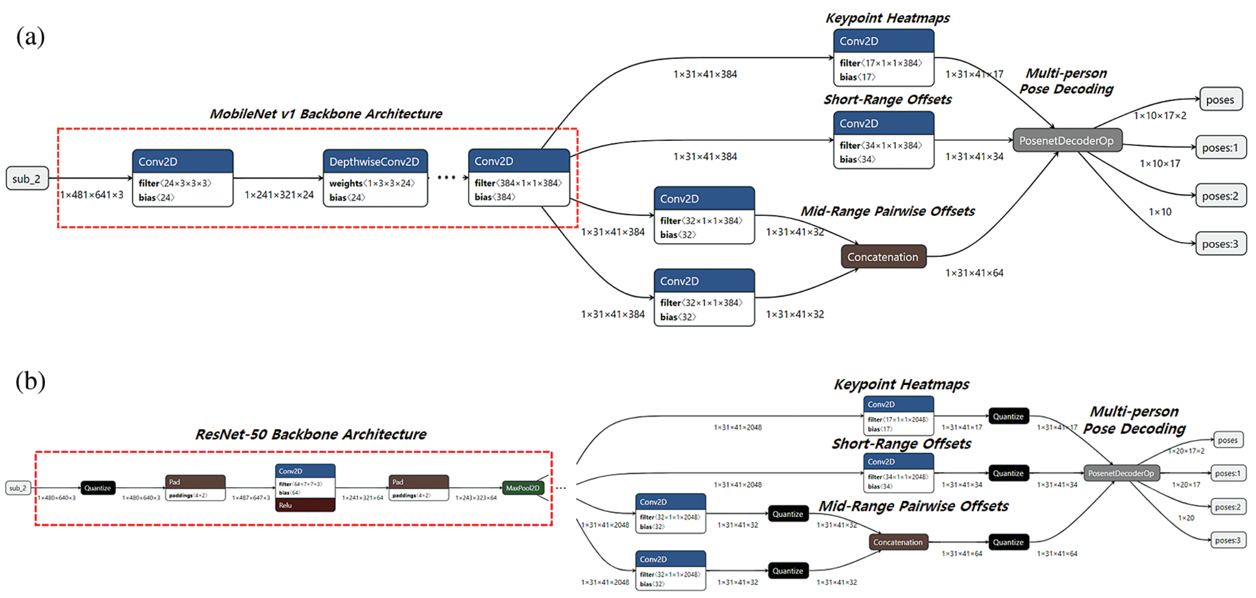


Figure 2: Comparison of PoseNet models with different backbone architectures (a) PoseNet model with MobileNet v1, (b) PoseNet model with ResNet-50

The MobileNet v1 consists of 18 Conv2D layers and 13 Depthwise Conv2D layers. The first Conv2D layer receives an image of (1 × 481 × 641 × 3) as the tensor of (batch, height, width, channels) as an input and outputs a feature map of (1 × 241 × 321 × 24) by using filters of (24 × 3 × 3 × 3) and a bias of (24). The second DepthwiseConv2D layer receiving the feature map outputs a feature map of (1 × 241 × 321 × 24) by using a weight of (1 × 3 × 3 × 24) and a bias of (24). In this way, other Conv2D and DepthwiseConv2D layers perform similar tasks repeatedly and the last Conv2D layer feed the same feature map of (1 × 31 × 41 × 384) to Conv2D layers for Keypoint Heatmaps, Short-Range Offsets and Mid-Range Pairwise Offsets. The Conv2D layers produce a feature map with 17 channels, a feature map with 34 channels, and a feature map with 64 channels, respectively. The PosenetDecoderOP for Multi-person Pose Decoding receives the feature maps as inputs and finally produces keypoint positions of (1 × 10 × 17 × 2), keypoint confidence scores of (1 × 10 × 17), and pose confidence scores of (1 × 10), when assuming that the maximum number of person instances detected is 10. Compared with the MobileNet v1, the ResNet-50 consists of 57 Conv2D and 4 MaxPool2D layers. The first Conv2D layer receives an image of (1 × 487 × 647 × 3) as an input and outputs a feature map of (1 × 241 × 321 × 64) by using filters of (64 × 7 × 7 × 3) and a bias of (64), the second MaxPool2D layer receiving the feature map outputs a feature map of (1 × 121 × 161 × 64), and so on. Also, the last Conv2D layer feed the same feature map of (1 × 31 × 41 × 2048) to Conv2D layers for Keypoint Heatmaps, Short-Range Offsets and Mid-Range Pairwise Offsets and the Conv2D layers produce feature maps which are fed to the PosenetDecoderOP. Thus, we will obtain keypoint positions of (1 × 20 × 17 × 2), keypoint confidence scores of (1 × 20 × 17), and pose confidence scores of (1 × 20), when assuming that the maximum number of person instances detected is 20.

Overall, we compare the number of Conv2D layers and parameters of two PoseNet models in Tab. 1. The number of parameters in each Conv2D layer can be calculated by

$$N_{Conv2D} = \{(\alpha \times \beta \times \gamma) + 1\} \times \delta \tag{6}$$

where α is the width of a filter, β is the height of a filter, γ is the number of filters in the previous layer, and δ is the number of filters. We use $\gamma = 1$ for each DepthwiseConv2D in MobileNet v1.

Table 1: Number of Conv2D layers and parameters in two PoseNet models

Type of backbone architectures	Number of layers	Number of parameters
MobileNet v1	31	1,180,147
ResNet-50	57	23,717,107

The model with MobileNet v1 has 31 layers and 1,180,147 parameters while the model with ResNet-50 does 61 layers and 23,717,107 parameters. In other words, computational complexity of the model with MobileNet v1 is about 20 times lower than that of the model with ResNet-50 because it has fewer number of layers and parameters, compared with the other. In terms of the computational complexity, therefore, the model with MobileNet v1 may be more suitable for the AIoT edge device than the model with ResNet-50. However, we need to analyze the performances of two models with regard to the pose estimation accuracy as well as the computation complexity and we will use the pose score as a performance metric of the pose estimation accuracy and the inference time and frame rate as performance metrics of the computational complexity in Section 3.

2.3 oneM2M-Compliant IoT Entities and Web Monitoring Application

The Floating Population Estimation block simply estimates the number of person instances by thinking of the number of pose confidence scores from the PoseNet model as it. Next, The IoT Client Application block modeled as an application dedicated node-application entity (ADN-AE) in oneM2M specifications [19] puts the temperature, humidity, illuminance, number of person instances data in a content instance and send the content instance to a remote IoT Server Middleware block considered as an infrastructure node-common service entity (IN-CSE) [20]. The IoT Server Middleware block stores the received content instance in the container of a Web Monitoring Application block modeled as an infrastructure node-application entity (IN-AE). Then it immediately notifies a message including all the data to the Web Monitoring Application block already permitted to subscribe it. The Web Monitoring Application block shows all the data on a map in a web browser periodically and this application may be expected to be used for various smart city services such as trade area analysis, crime prevention, etc.

3 Experimental Results

We implemented the AIoT edge device by using a Raspberry Pi 4 Model B with a Google Coral USB Accelerator. The Google Coral USB Accelerator provides high-performance deep learning inference at low power costs. To run this, we installed PyCoral, TensorFlow Lite Runtime, and libedgetpu-max [21]. In addition, we installed a DHT11 sensor for detecting temperature and humidity, a BH1750 sensor for detecting illuminance in Lux, and a C270 HD webcam for capturing real-time videos at 30 frames per second (fps) on the AIoT edge device. For the purpose of comparing performances of PoseNet models with MobileNet v1 and ResNet-50 architectures, we pre-recorded three 50 second video clips with 480 × 360, 640 × 480, and 1280 × 720 resolutions as inputs and considered inference time, frame rate, and pose score as performance metrics. Moreover, to implement the IoT client application and server middleware, we exploited oneM2M-compliant IoT platforms called &Cube and Mobius, respectively [22]. For implementing the floating population monitoring server, Kakao map web application programming interface (API) [23] was used to display temperature, humidity, illuminance, and floating population estimation results from the AIoT edge device.

We compare inference times of two PoseNet models on the AIoT edge device according to video resolutions in Fig. 3. For 480 × 360 video clip, the model with MobileNet v1 shows inference time of 8.19 milliseconds (ms) and the other with ResNet-50 shows that of 66.8 ms. For 640 × 480 video clip, the model with MobileNet v1 shows inference time of 13.89 ms and the other with ResNet-50 shows that of 537.6 ms. For 1280 × 720 video clip, the model with MobileNet v1 shows inference time of 43.51 ms and the other with ResNet-50 shows that of 1203 ms. In other words, the model with MobileNet v1 can estimate human poses about 27 times faster than the other with ResNet-50 because it has fewer number of parameters and smaller computational complexity than the model with ResNet-50. As the video resolution increases, the difference between their inference times increases rapidly. In terms of inference time, we can say that the model with ResNet-50 is very sensitive to the video resolution.

Fig. 4 shows frame rates, that refer to speeds at which the AIoT edge device displays the video clip including pose estimation results on a single screen, of two PoseNet models. For 480 × 360 video clip, the model with MobileNet v1 provides 122.5 fps that is about 8 times faster than the other model with the ResNet-50. For 640 × 480 video clip, the model with MobileNet v1 provides 70.37 fps while the other model with ResNet-50 does 1.86 fps. For 1280 × 720 video clip, the model with MobileNet v1 provides 23.04 fps while the other model with ResNet-50 does 0.82 fps. In terms of frame rate, the model with ResNet-50 is not suitable for use on the AIoT edge device because it only supports frame rates less than 30 fps. Also, the model with MobileNet v1 cannot display the video clip smoothly if an input video clip with 1280 × 720 resolution is fed into it.

In Fig. 5, pose scores of two PoseNet models are shown according to video resolutions. The pose score or pose confidence score denotes the mean of confidence scores of 17 keypoints or heatmap positions detected by the PoseNet models. As video resolution increases, their pose scores slightly decrease. The difference between their pose scores seems negligible and they can offer over 0.7 pose scores regardless of the video resolution. After analyzing all the results from Figs. 3–5, we found that the model with MobileNet v1 is more suitable for our AIoT edge device than the model with ResNet-50 since it can estimate human poses very fast without significant pose score degradation. Video resolutions of 480 × 360 and 640 × 480 pixels are recommended as inputs to the model with MobileNet v1.

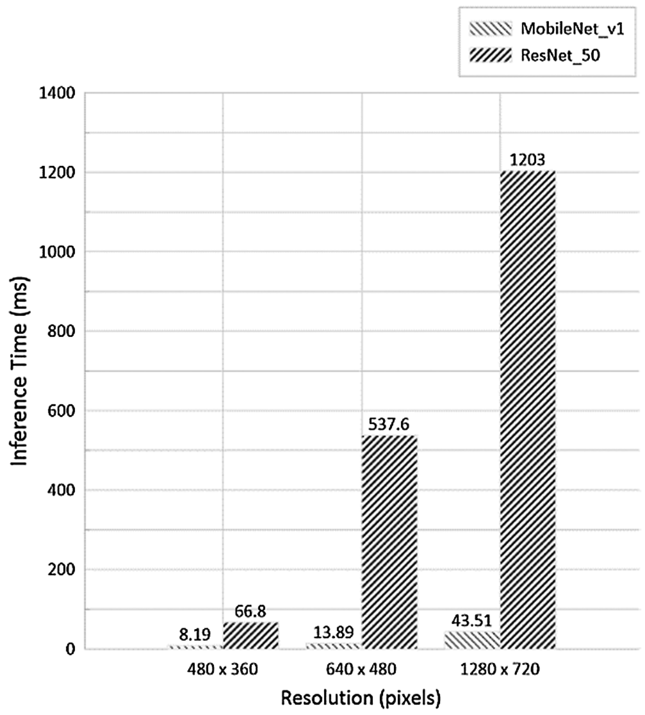


Figure 3: Inference times of PoseNet models based on MobileNet v1 and ResNet-50 architectures according to input video clips with three different resolutions

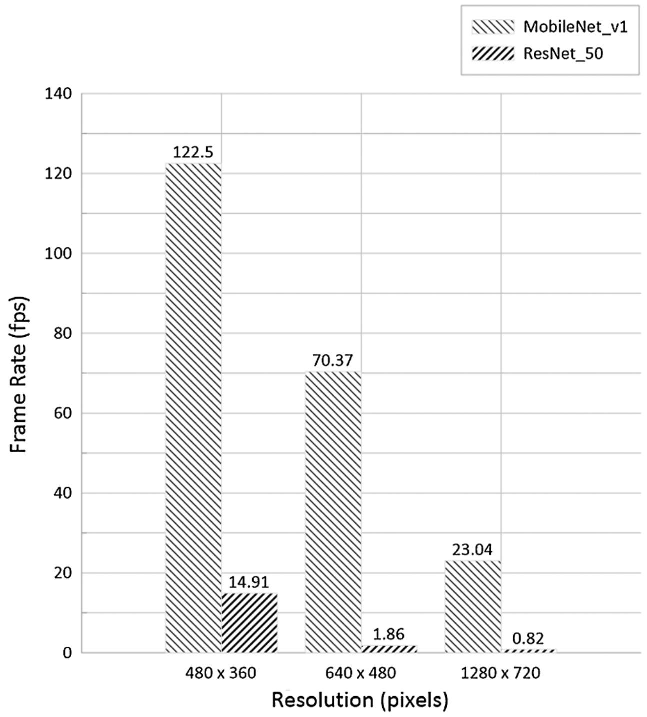


Figure 4: Frame rates of PoseNet models based on MobileNet v1 and ResNet-50 architectures according to input video clips with three different resolutions

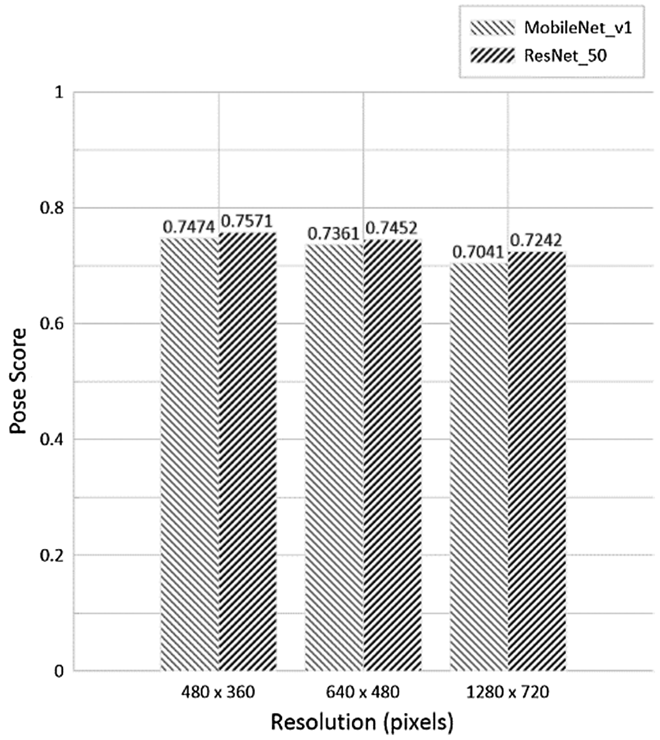


Figure 5: Pose scores of PoseNet models based on MobileNet v1 and ResNet-50 architectures according to input video clips with three different resolutions

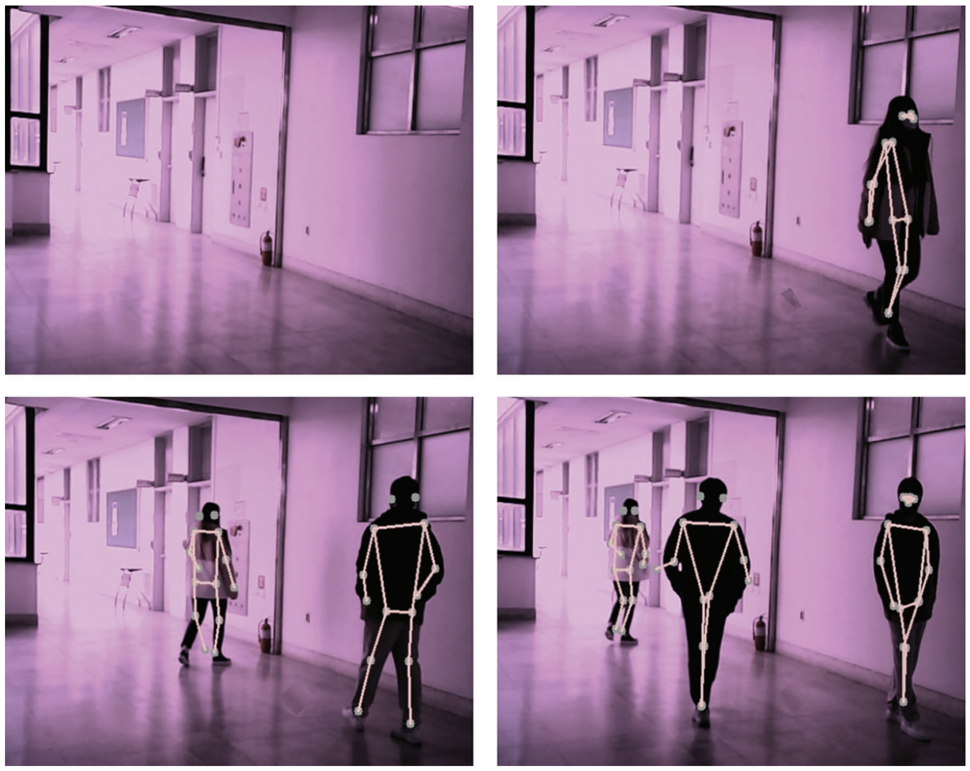
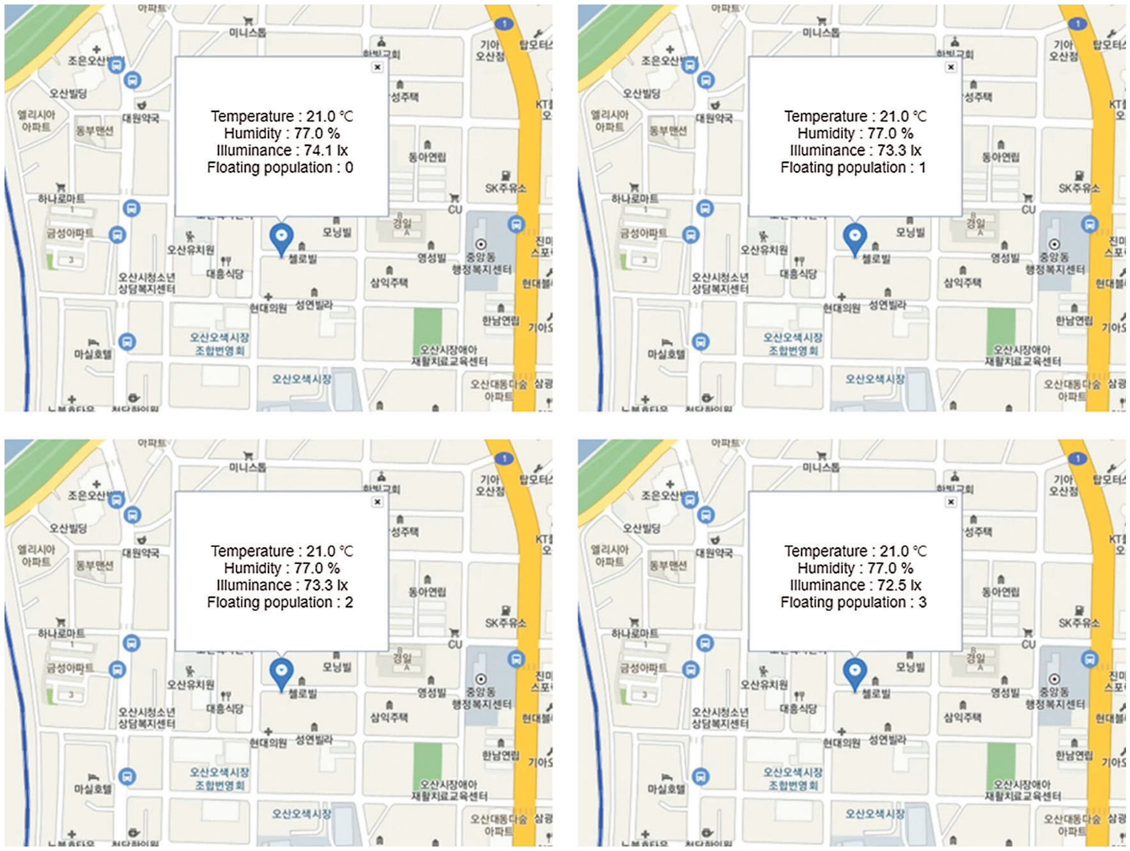


Figure 6: The output of the PoseNet model with MobileNet v1 for a real-time video from the C270 HD webcam installed on the AIoT edge device

In Fig. 6, a captured output image of the PoseNet model using MobileNet v1 is shown for a real-time video from the C270 HD webcam installed on the AIoT edge device where all the operations of the model were verified. Finally, we verified the interaction between oneM2M-compliant platforms and a web monitoring application and displayed all the information about temperature, humidity, illuminance, and floating population through the Kakao map web API in Fig. 7.



Temperature : 21.0 ℃
Humidity : 77.0 %
Illuminance : 74.1 lx
Floating population : 0

Temperature : 21.0 ℃
Humidity : 77.0 %
Illuminance : 73.3 lx
Floating population : 1

Temperature : 21.0 ℃
Humidity : 77.0 %
Illuminance : 73.3 lx
Floating population : 2

Temperature : 21.0 ℃
Humidity : 77.0 %
Illuminance : 72.5 lx
Floating population : 3

Figure 7: The outputs of the interaction between oneM2M-compliant platforms and a web browser through the Kakao map web API: temperature, humidity, illuminance, and floating population

4 Conclusion

In this paper, we presented an oneM2M-compliant system including an AIoT edge device with two PoseNet models based on MobileNet v1 and ResNet-50 backbone architectures and selected the PoseNet model with MobileNet v1 suitable for the AIoT edge device by comparing the models in terms of inference time, frame rate and pose score. Experimental results showed that the model with MobileNet v1 could estimate human poses very fast without significant pose score degradation when compared with the model with ResNet-50. In other words, the difference between their pose scores was negligible but the inference time of the model with MobileNet v1 was faster than that of the model with ResNet-50 regardless of video resolutions. In terms of frame rate, video resolutions of 480 × 360 and 640 × 480 pixels are recommended as inputs to the model with MobileNet v1. Finally, the overall oneM2M-compliant system applicable to commercial area analysis in a smart city was implemented. It consisted of an AIoT edge device containing the model with MobileNet v1 and a floating population monitoring server and periodically displayed temperature, humidity, illuminance and floating population estimation data on a map in a web browser.

Funding Statement: This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2018-0-00387, Development of ICT based Intelligent Smart Welfare Housing System for the Prevention and Control of Livestock Disease).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Varghese, S. K. Vargheese and E. Peter, "A study on artificial intelligence of things: Techniques and applications," *A Journal of Composition Theory*, vol. 13, pp. 888–896, 2020.

[2] R. Revathy, M. G. Raj, M. Selvi and J. K. Periasamy, "Analysis of artificial intelligence of things," *International Journal of Electrical Engineering and Technology*, vol. 11, no. 4, pp. 275–280, 2020.

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

[4] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.

[5] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[6] C. Gong, F. Lin, X. Gong and Y. Lu, "Intelligent cooperative edge computing in internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9372–9382, 2020.

[7] S. B. Calo, M. Touna, D. C. Verma and A. Cullen, "Edge computing architecture for applying AI to IoT," in *Int. Conf. on Big Data*, Boston, MA, USA, pp. 3012–3016, 2017.

[8] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo *et al.,* "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[9] W. Sun, J. Liu and Y. Yue, "AI-enhanced offloading in edge computing: When machine learning meets industrial IoT," *IEEE Network*, vol. 33, no. 5, pp. 68–74, 2019.

[10] J. Tang, D. Sun, S. Liu and J. Gaudiot, "Enabling deep learning on IoT devices," *Computer*, vol. 50, no. 10, pp. 92–96, 2017.

[11] Z. Cao, G. Hidalgo, T. Simon, S. E. Wei and Y. Sheikh, "Openpose: Realtime multi-person 2D pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 172–186, 2021.

[12] J. Rivera, "Practical TensorFlow.js," in *Deep Learning in Web App Development*, 1st. edition, Berkeley, CA, USA: Apress, pp. 126–149, 2020.

[13] H. J. Park and K. Lee, "Implementation of an open artificial intelligence platform based on web and tensorflow," *Journal of Information and Communication Convergence Engineering*, vol. 18, no. 3, pp. 176–182, 2020.

[14] A. Ghorai, S. Gawde and D. Kalbande, "Digital solution for enforcing social distancing," in *Proc. ICICC*, New Delhi, India, 2020.

[15] F. Rishan, B. De Silva, S. Alawathugoda, S. Nijabdeen, L. Rupasinghe *et al.,* "Infinity yoga tutor: Toga posture detection and correction system," in *Int. Conf. ICITR*. Moratuwa, Sri Lanka, pp. 1–6, 2020.

[16] T. Y. Ha and H. J. Lee, "Analysis on the mobile healthcare behavior using an artificial intelligence based pose estimation," *Journal of the Institute of Electronics and Information Engineers*, vol. 57, no. 1, pp. 63–69, 2020.

[17] G. Papandreou, T. Zhu, L. C. Chen, S. Gidaris, J. Tompson *et al.,* "Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model," *Lecture Notes in Computer Science*, vol. 11218, pp. 282–299, 2018.

[18] X. Jia, Y. Xiao, D. Yang, Z. Yang and C. Lu, "Multi-parametric MRIs based assessment of hepatocellular carcinoma differentiation with multi-scale ResNet," *KSII Transactions on Internet and Information Systems*, vol. 13, no. 10, pp. 5179–5196, 2019.

[19] I. Y. Ahn, N. Sung, J. Lim, J. Seo and I. D. Yun, “Development of an oneM2M-compliant IoT platform for wearable data collection,” *KSII Transactions on Internet and Information Systems*, vol. 13, no. 1, pp. 1–15, 2019.

[20] oneM2M, *Service layer core protocol specification,* Release 2 TS-0004 v2.27.0, 2020.

[21] Coral. [Online]. Available: https://coral.ai/software/.

[22] OCEAN, Mobius Release 2 v2.0. 0, Installation Guide, 2018.

[23] R. Rijayanti, R. F. Muhammad and M. Hwang, “Vehicle waiting time information service using vehicle object detection at fuel charging station,” *Journal of Information and Communication Convergence Engineering*, vol. 18, no. 3, pp. 147–154, 2020.