Performance Evaluation of Edge Computing-Based Deep Learning Object Detection

Chuan-Wen Chen, Shanq-Jang Ruan, Chang-Hong Lin, Chun-Chi Hung
National Taiwan University of Science and Technology, Department of Electrical & Computer Engineering
No.43, Keelung Rd., Sec.4, Da'an Dist.,
Taipei City 10607, Taiwan (R.O.C.)
+886-2-2737-6411
{m10502145, sjruan, chlin, m10602125}@mail.ntust.edu.tw

ABSTRACT
This article presents a method for implementing the deep learning object detection based on a low-cost edge computing IoT device. The limit of the hardware is a challenge for working the pre-trained neural network model on a low-cost IoT device. Hence, we utilize the Neural Compute Stick (NCS) to accelerate the neural network model on a low-cost IoT device by its high efficiency floating-point operation. With the NCS, the low-cost IoT device can successfully work the pre-trained neural network model and become an edge computing device. The experimental results show the proposed method can effectively detect the objects based on deep learning on an edge computing IoT device. Furthermore, the objective experiment demonstrates the proposed method can immediately infer the neural network model for images in average 1.7 seconds with only one of the NCS and the neural network model can reach average 9.2 fps for the video sequences with four NCSs acceleration. In addition, the discrepancy of the neural network model between the edge device and the edge server is less than 2% mean average precision (mAP).

CCS Concepts
• Computing methodologies→Object detection

Keywords
Object Detection; Edge Computing; Deep Learning.

1. INTRODUCTION
Over the course of the past ten years, deep learning, artificial intelligence, and IoT are developed rapidly with the progress of technologies. Meanwhile, this means that the data and the computation overhead have also increased significantly with the extension of artificial intelligence applications. The data can be placed in the cloud system and analyzed through the cloud computing [1]. However, all the cloud system centers are located in the specific geological locations. If the data analysis is performed over a long distance, the network latency will hard to be avoided and will also be unable to reach the requirement for the immediate response. As a result, the concept of fog computing is derived [2], which is a distributed infrastructure can create the low-latency network connections between the device and the analytics endpoints. Certain of the application processes or services are managed at the device, but others are still managed in the cloud. Therefore, the fog computing is much closer to the users than the cloud computing. However, the computation overhead of the deep learning is extremely large, which can not process by fog computing.

ICNCC 2018, December 14–16, 2018, Taipei City, Taiwan

ACM ISBN 978-1-4503-6553-6/18/12…$15.00
https://doi.org/10.1145/3301326.3301369

In order to achieve the deep learning on IoT devices and reduce the network latency caused by data transmission, the concept of edge computing is proposed [3]. The characteristics of the edge computing are real-time and capable of artificial intelligence, e.g., self-driving car, face recognition drone, and augmented reality [4]. Another advantage of the edge computing is protecting users' information privacy. For examples, the personal information or the medical information are related to the relevant laws. It is more necessary to protect this type of information. If the edge computing can be completed on end-devices, it can effectively reduce the probability of hacking and protect the privacy of users. However, there are still many other problems to reach the deep learning on edge computing IoT device. For instance, whether the deep learning neural network models can implement on low-cost IoT device or utilize the high-cost CPU/GPU to process the large computation overhead of the IoT device in the deep learning edge computing. Therefore, it is essential to consider the trade-off between the cost and the performance.

In this article, we describe a method for implementing deep learning object detection on a low-cost edge computing IoT device. To achieve the deep learning on a low-cost IoT device, we utilize the Neural Compute Stick (NCS) to accelerate the neural network model on a low-cost IoT device through its high-performance operation. The proposed method is trained and verified by the two public datasets - PASCAL VOC [5] and KITTI [6]. Besides above datasets, we have built our dataset LPS2017 for training and verifying the proposed method. The experiment illustrates that the proposed method can promptly infer the neural network model for images in average 1.7 seconds with one NCS. Additionally, we adopt the distributed computing method with four NCSs to accelerate the neural network model. Our results indicate that with accelerated by four NCSs, the dynamic videos can realize average with 9.2 fps. Furthermore, regarding the discrepancy between the edge device and the edge server, which are less than 2% mAP for the same neural network model.

The remainder of this article is organized as follows. Section 2 presents the related literature by employing deep learning on edge computing devices. Section 3 thoroughly elaborates the flow of

the proposed method. The experimental results are delivered and discussed in Section 4. Section 5 is the conclusion of this article.

2. LITERATURE REVIEW

As field of research, there have been rising interest in using deep learning on the edge computing device in the recent literature, e.g., [7-10]. Liu et al. [7] proposed a method for deep learning food recognition on an edge computing service. At first, the food image is captured by the mobile device. Then the image filter is employed, and the image is segmented on the mobile device. Furthermore, the preprocessed image is delivered to the PC server to identify the food by the pre-trained GoogleNets. Finally, the result is transmitted to the mobile device so that the user can obtain the food information. In [8], they proposed a deep learning edge computing system based on the Electroencephalogram (EEG) signal to monitor and evaluate the effects of epilepsy on the brain. EEG signal is generated by using the wearable sensors and delivered to the cloud server to classify by employing a pre-trained CNN model. The results are passed back to the edge device for evaluation. Ren et al. [9] proposed a deep learning object detection system based on the edge computing for surveillance applications. The proposed system illustrates that the region of interest is extracted from the input image by the edge device and delivered to the PC server for detecting objects through the pre-trained CNN model. The result is delivered to the edge device for the user. In [10], in order to implement the deep learning on the edge computing IoT device, the optimization method is proposed. To enhance the fault tolerance of the deep learning architecture, the typical method is to quantize the floating-point number from 32 bits to 1-9 bit(s). This type of quantization not only decreases the computation overhead but also reduces the memory usage of data access. The quantified recognition only reduced 2.9%, which is a commendable optimization method to improve fault tolerance of deep learning architectures.

The common problem of the above methods [7-9] is that they can not directly infer the neural network model on the IoT device itself. All of them have to complete the model inference in the server and deliver the results back to the edge device, which causes the delay to the data transmission and privacy issue. Therefore, in our proposed method, we aim to how to easily implement the deep learning object detection on a low-cost IoT device and assess the performance.

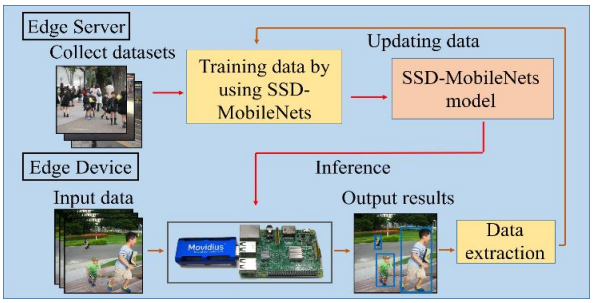

Figure 1. Flow diagram of the proposed method.

3. THE PROPOSED METHOD

In this section, we first describe our proposed method. Figure 1 presents the flow diagram of the proposed method. The flow is divided into two parts, namely the edge server and the edge device. In the edge server, which is for training SSD-MobileNets [11, 12]. The training is on the datasets, PASCAL VOC [5], KITTI [6], and LPS2017. And then the SSD-MobileNets trained by the edge server is compiled and delivered to the edge device. In order to successfully infer the SSD-MobileNets model in the edge device, we use the NCS to infer and accelerate the SSD-MobileNets model. To verify the object detection accuracy of the SSD-MobileNets model, we employ the webcam on the edge device to obtain the real-time stream and also utilize the verification datasets. At the same time, the results of the detection are extracted and transmitted to the edge server to train the SSD-MobileNets model again for updating the model weights to improve the precision accuracy.

3.1 SSD-MobileNets

Single Shot MultiBox Detector (SSD) is one of the deep learning object detection methods [11]. Compared to the other deep learning object detection methods, such as R-CNN [13], Fast RCNN [14], and YOLO [15], SSD is the fastest and the most accurate in the deep learning object detection. To optimize the training, the SSD ignores the proposals generation, pixel resampling, and feature resampling, besides, the SSD employs the multiscale feature maps for detection. MobileNets is an efficient neural network model for mobile and embedded vision applications. MobileNets architecture is based on depthwise separable convolutions which is combined with a depthwise convolution and a pointwise convolution. Depthwise convolution applies a single filter to each input feature maps channel. And then the pointwise convolution utilizes the 1x1 convolution to combine the outputs of the depthwise layer. By using MobileNets, it is 9 times less computation overhead than the standard convolution at only a slight reduction in accuracy. Therefore, we replace the original VGG-16 network architecture [16] in SSD with MobileNets.

3.2 Datasets Collection

For detection of the object categories, there are many different datasets for adoption. In this article, we integrate the PASCAL VOC [5] for our training datasets. The PASCAL VOC contains 20 object categories and spreads over 16,000 images and also over 40,000 objects are labeled. KITTI is the largest autonomous driving scene dataset in the world [6]. It contains the 7 object categories and distributes over 7,000 images with over 80,000 objects are labeled. In addition to above two public datasets for training and verifying, we have built our datasets LPS2017 for test. The LPS2017 contains the adult and the child two categories. It spreads over 6,500 images and also over 35,000 objects are labeled.

3.3 Model Inference

The SSD-MobileNets adopts the supervised learning for training [17]. The supervised learning is one of the learning methods, which has to label the corresponding labels before training the neural network model. For example, the model is trained by 1000 images which are labeled with dogs and cats, and then the model can recognize whether the dogs or the cats with a new image. In our proposed method, we employ the Caffe framework [18] for training, which is one of the deep learning frameworks developed by Berkeley AI Research (BAIR). After the model are trained by the Caffe framework on the edge server, we have to implement the caffe model on the edge device. In this article, we utilize the Raspberry Pi 3 as our edge device. Due to its Wi-Fi and Bluetooth capabilities, which are easier for users to develop the IoT applications. Though there already exists a lot of API for users to infer the pre-trained neural network model for the deep learning object detection on the Raspberry Pi 3, the inference time for detection costs 30 seconds per image. It is a disparity for

immediately infer the deep learning object detection. Consequently, we use the NCS to accelerate and infer the SSD-MobileNets on the Raspberry Pi 3. The NCS is a tiny fan-less deep learning device with high efficiency floating-point operation, which can be used to infer neural network model and learn AI programming at the edge device. Figure 2 shows the component of the edge device in this article.

Table 1. Setting of experimental platforms

Device	Hardware	Price	Power Consumption	Usage
Edge device	Raspberry Pi 3, Neural Compute Stick(s), Logitech C310 webcam	$150	10W	Model inference
Edge server	(PC) i5-7400, GTX 1080, 16GB RAM	$1,500	300W	Model training

Table 2. The accuracy results of the PASCAL VOC 2007 validation dataset

	mAP	Person	Bird	Cat	Cow	Dog	Horse	Sheep	Aero	Bike	Boat	Bus	Car	Motor	Train	Bottle	Chair	Table	Sofa	TV	Plant
PC (%)	64.59	69.2	61	78	71	78.5	78.3	61.5	70.4	70.9	46.3	72	70.5	70.8	78.5	37.6	47.2	56.8	65.1	69.3	38.7
Pi+NCS (%)	64.24	69	60	77.8	70.5	78	78.1	61	70.1	70.8	46.3	71	70.5	70.7	78.5	37.5	47	55.9	64.9	69	38.1

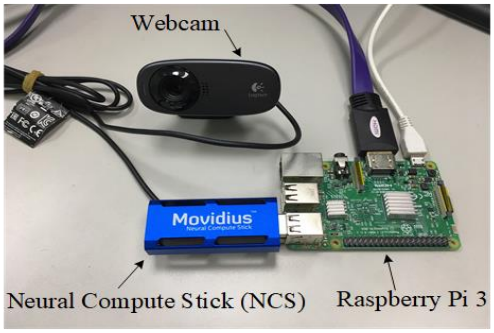

Figure 2. The component of the edge device.

Table 3. The accuracy results of the KITTI validation dataset

	mAP	Car	Van	Truck	Walkers	Person	Cyclist	Tram
PC (%)	57.83	62.4	58.5	59.6	60.1	55.7	50.6	57.9
Pi+NCS (%)	56.86	62.1	57.5	59.1	58.9	54.6	49.9	55.9

Table 4. The accuracy results of the LPS2017 validation dataset

	mAP	Adult	Child
PC (%)	61.3	64.2	58.4
Pi+NCS (%)	59.4	62.4	55.8

4. EXPERIMENTAL RESULT

In this section, we describe the exhaustive evaluation of our proposed method. The experiments are performed on the edge device and the edge server. Table 1 shows the component of the edge device and the edge server respectively.

4.1 Object Detection Accuracy

To evaluate the accuracy of the proposed scheme, the mean average precision (mAP) is employed, which is the metric to measure the accuracy of the deep learning object detection. The higher of the mAP value, the more accurately the object is detected. We adopt the PASCAL VOC, KITTI, and the LPS2017 for verifying the accuracy. Table 2 to Table 4 show the results of the three different validation datasets. Although the cost of the edge server (PC) is higher than the edge device (Pi+NCS), it is obviously that the overall mAP are less than 2% mAP between the edge device (Pi+NCS) and the edge server in each validation datasets.

4.2 Inference Execution Time

In order to assess the performance of the inference execution time, execution time of the images and the videos are evaluated respectively. It is uncomplicated to employ one NCS for inferring the SSD-MobileNets with high efficiency floating-point operation on the edge device. To further accelerate the inference time, the distributed computing method is utilized for the multiple NCSs as shown in Fig. 3. Rather than using the multiple NCSs for accelerating the inference time on same frame, each NCS is correspond to one input frame for accelerating. Therefore, the video sequence is more suitable for accelerating than the images.

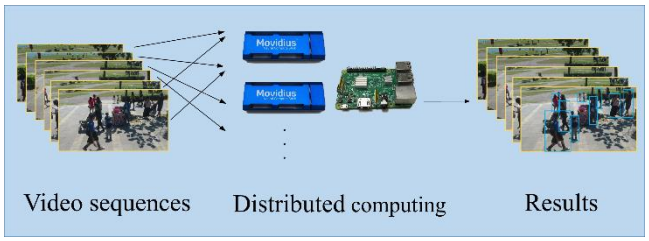

Figure 3. Distributed computing method.

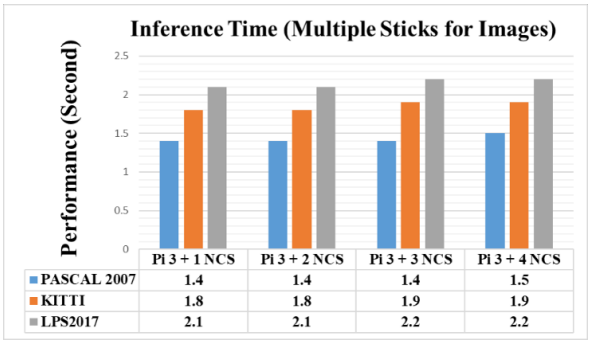

Figure 4. Inference time for images.

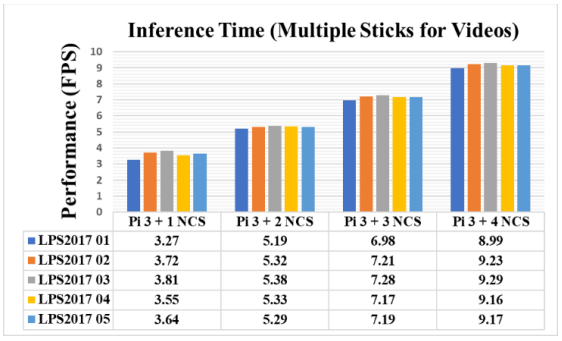

Figure 5. Inference time for videos.

Figure 4 shows the results of the inference time for images, which are inferred by the multiple NCSs. As shown in Fig. 4, it only cost average 1.7 seconds by using one NCS to accelerate, which reached the requirement of the immediate response. In addition, we compare the four different types of acceleration, which contains one to four NCSs for accelerating. By adopting the distributed computing method, it is no evident disparity to accelerate between the four different types.

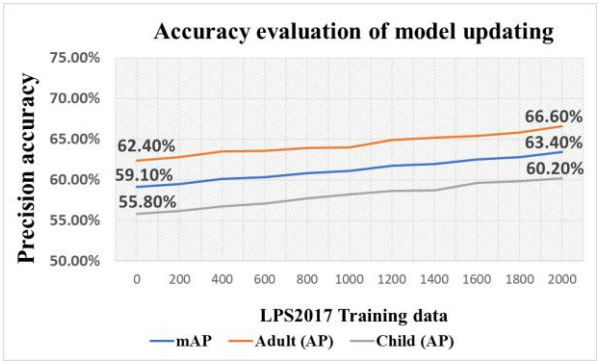

Figure 6. Accuracy evaluation of model updating.

The results of the inference time for the video sequences, which are inferred by the multiple NCSs are presented in Fig. 5. In Fig. 5, the four different types for accelerating are also employed. Through the distributed computing method, the fps of four NCSs acceleration are 2.5 times faster than that of using one NCS. At present, the maximum number for the parallel accelerating are four NCSs. Consequently, the growth curve of the fps becomes smoother as the number of increasing the NCS.

4.3 Model Updating Accuracy

Besides assessing the performance of the inference time, on the other hand, the accuracy of the model updating is presented in this subsection. Figure 6 shows the results of the updating precision accuracy. We extract the result images of the object detection to increase the training data for training the model again, which can update the model weights. As shown in Fig. 6, we add the 200 images for training in each time and also assess the precision accuracy. The accuracy of updating model increases 4.2% via our experiment.

5. CONCLUSION

In this article, a method for inferring SSD-MobileNets based on low-cost edge computing IoT device is demonstrated. By exploiting the NCS, we can successfully implement SSD-MobileNets on the Raspberry Pi 3, which is trained by the edge server. The proposed method is verified by the three datasets (PASCAL VOC, KITTI, and LPS2017). The experiment results show that the proposed method can infer image on SSD-MobileNets in 1.7 seconds with one NCS and the acceleration for the video sequences can reach 9.2 fps with four NCSs. Moreover, the accuracy divergence between the edge device and the edge server are less than 2% mAP for the same neural network model.

6. REFERENCES

[1] S. Patidar, D. Rane and P. Jain. 2012. A Survey Paper on Cloud Computing. In *2nd International Conference on Advanced Computing & Communication Technologies*, pp. 394–398, Jan. 2012.

[2] S. Yi, Z. Hao and Q. Li. Fog Computing: Platform and Applications. *In Processing of IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, pp. 73-78, Nov. 2015.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu. Edge Computing: Vision and Challenges. In *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637-646, Oct. 2016.

[4] R. T. Azuma. A Survey of Augmented Reality. In *Presence: Teleoperators and Virtual Environments*, vol. 6, no. 4, pp. 355–385, 1997.

[5] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn and A. Zisserman. The PASCAL Visual Object Classes Challenge: A Retrospective. In *International Journal of Computer Vision*, vol. 111, no.1, pp. 98-136, Jan. 2015.

[6] G. Andreas, L. Philip, S. Christoph and U. Raquel. Vision meets robotics: The KITTI dataset. in *International Journal of Robotics Research*, pp. 1231-1237, Sept. 2013.

[7] C. Liu, Yu. Cao, Y. Luo, G. Chen, V. Vokkarane, M. Yunsheng, S. Chen and P. Hou, ”A New Deep Learning-Based Food Recognition System for Dietary Assessment on an Edge Computing Service Infrastructure,” in IEEE Transactions on Services Computing, vol. 11, no. 2, pp. 249-261, Jan. 2017.

[8] M. Hosseini, T. Tran, D. Pompili, K. Elisevich and H. Soltanian- Zadeh, ”Deep Learning with Edge Computing for Localization of Epileptogenicity using Multimodal rs-fMRI and EEG Big Data,” in IEEE International Conference on Autonomic Computing, pp. 83-92, Jul. 2017.

[9] J. Ren, Y. Guo, D. Zhang, Q. Liu and Y. Zhang, ”Distributed and Efficient Object Detection in Edge Computing: Challenges and Solutions,” in IEEE Network, no. 99, pp. 1-7, Apr. 2018.

[10] M. Verhelst, and B, Moons, ”Embedded Deep Neural Network Processing: Algorithmic and Processor Tech-niques Bring Deep Learning to IoT and Edge Devices,” in IEEE Solid-State Circuits Magazine, vol. 9, no. 4, pp. 55-65, Nov. 2017.

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu and A. C. Berg, ”SSD: Single Shot Multibox Detector,” in European Conference on Computer Vision, pp. 21–37, Springer, 2016.

[12] A. Howard, M. Zhu, B. Chen and D. Kalenichenko, ”MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” arXiv preprint arXiv :1704.04861, 2017.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, ”Rich feature hierarchies for accurate object detection and semantic segmentation,” in Proc. Conference on Computer Vision and Pattern Recognition (CVPR), pp. 580-587, Jan. 2014.

[14] R. Girshick, ”Fast R-CNN,” in IEEE International Conference on Computer Vision (ICCV), pp. 1440-1448, Dec. 2015.

[15] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, ”You only look once: unified, real-time object detection,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779-788, Jun. 2016.

[16] K. Simonyan and A. Zisserman, ”Very Deep Convolutional Networks for Large-scale Image Recognition,” CoRR, abs/1409.1556, pp. 249-268, 2007.

[17] S. B. Kotsiantis, ’Supervised machine learning: A review of classification techniques,” Informatica , 2014.

[18] Y. Jia, E. Shelhamer et al., ”Caffe: Convolutional architecture for fast feature embedding,” in Proceedings of the 22nd ACM International Conference on Multimedia, pp. 675-678,