The Detailed Data on the Neural Compute Stick Acceleration Performance

Qian Li
State Key Laboratory of Acoustics, Institute of Acoustics
Chinese Academy of Sciences
Beijing, China
University of Chinese Academy of Sciences
Beijing, China
liqian7788@163.com

Jiayu Song
School of Computer Science and Engineering
Beihang university
Beijing, China
jiayu_buaa@163.com

Jiangbo Ning
State Key Laboratory of Acoustics, Institute of Acoustics
University of Chinese Academy of Sciences
Beijing, China
ningjiangbo@163.com

Jianping Yuan
Institute of Acoustics
Chinese Academy of Sciences
Beijing, China
yuanjianping@163.com

***Abstract*—The four trends such as real-time, intelligence, security and privacy have spawned the rise of edge computing and front-end intelligence. The Neural Computing stick(NCS) can move calculations from the cloud to the terminal, and provide dedicated deep neural network acceleration for terminal AI devices. However, there is currently no detailed data comparison of the NCS acceleration performance. This paper compares the computational speed of convolutional neural networks on the Raspberry Pi in the presence or absence of the NCS. In addition, we further compare the calculated speeds of the CPU, GPU, and NCS through the vehicle model. The result shows that the NCS can achieve 4 to 6 times acceleration in neural networks.**

***Index Terms*—NCS; network acceleration; convolutional neural networks; edge computing**

I. Introduction

With the rise of artificial intelligence, various algorithms for deep learning have been explored one after another. The Convolutional Neural Network is one of the representative algorithms of deep learning, which is widely used in computer vision, natural language processing and other fields [1]. R-CNN [2] is first used in the field of target detection. Faster R-CNN [3] achieves real-time object detection. YOLO [4] further improves the performance of the algorithm. However, the existing AI algorithms rely too much on expensive and cost-inefficient GPU servers for cloud processing. The intelligent processing is limited by the transmission bandwidth and cannot be handled intelligently in real time. Meanwhile, the transmission of massive data exacerbates the load on the network and servers. With the advent of the ultra-low latency and ultra-reliable 5G era, edge computing is a new trend. Processing data at the edge of a network enables the execution of data processing functionalities in proximity to the physical inputs and outputs while overcoming significant drawbacks of a centralized computing approach [5]. The main benefits of Edge Computing (EC) are an increase in reaction speed due to the reduction of latency, an increase of data protection andsovereignty, processing of high data quantities without the need for quick storage and the possibility to operate autonomously from the cloud [6]. As the developing trend in edge computing, it is an important direction for implementing high-performance and low-power consumption of hardware.

Raspberry Pi is a typical hardware for edge computing, which is small, powerful, cheap, hack-able [7]. Meanwhile, it would be excellent if we further increase its calculation speed. Graphics processor (GPU) is good at large-scale parallel computing. It provides a vast number of simple, data-parallel, deeply multi-threaded cores and high memory bandwidths. GPU architectures are becoming increasingly programmable, offering the potential for dramatic speedups for a variety of general-purpose applications compared to contemporary general-purpose processor (CPU) [8]. However, some unfavorable factors limit its application in low-power AI terminals, such as high price, large size and high power consumption. The Neural Computing stick(NCS) effectively meets the need of high-performance and low-power consumption. It is a USB-based deep learning inference tool and has more video processing units(VPU). There are three typical application scenarios of artificial intelligence in edge computing: image recognition, 3D modeling and indoor navigation. The common feature of these applications is the processing of video. With many video processing units, the NCS can provide dedicated deep neural network acceleration for a wide range of mobile and embedded vision devices.

In order to provide the detailed data comparison for the NCS acceleration performance, this paper compares the computa-

The national natural science foundation of China(61601454)

978-1-7281-4094-0/19/$31.00 ©2019 IEEE

tion speed of convolutional neural networks on the CPU, GPU and NCS. This study is as follows,(1)From the perspective of loading the model and detecting the picturewe compare the speed of the SqueezeNet model on the Raspberry Pi in the presence or absence of the NCS.(2)We contrast the NCS directly with the CPU and GPU of the host by running the vehicle and license plate detection model in the same picture. The result show that the NCS can achieve 4.71 acceleration in SqueezeNet, and 5.15x acceleration in AlexNet, 6.47x acceleration in GooleNet, which is more suitable for a wide range of mobile and embedded vision devices.

II. NETWORK STRUCTURE

A. SqueezeNet architecture

SqueezeNet [9] is a miniaturized network architecture proposed by Forrest N. Iandola and Song Han, which achieves AlexNet-level accuracy on ImageNet with 50x fewer parameters. Additionally, with model compression techniques, it compresses the original AlexNet to about 510 times without sacrificing accuracy, which makes it more feasible to deploy on FPGAs and other hardware with limited memory.

The core of the SqueezeNet network is the Fire module shown in Fig.1. A Fire module is comprised of: a squeeze convolution layer(which has only 1x1 filters), feeding into an expand layer that has a mix of 1x1 and 3x3 convolution filters.

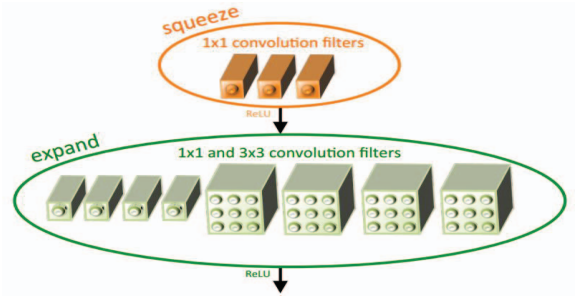

Fig. 1. Macroarchitectural view of SqueezeNet architecture. Left: SqueezeNet. Middle: SqueezeNet with simple bypass. Right: SqueezeNet with complex bypass.

The macroarchitectural view of SqueezeNet architecture is shown in Fig.2. SqueezeNet begins with a standalone convolution layer (conv1), followed by 8 Fire modules (fire2-9), ending with a final convolution layer (conv10). SqueezeNet performs max-pooling with a stride of 2 after layers conv1, fire4, fire8, and conv10.

B. Structure introduction

The network structure, as shown in Fig.3, begins with a standalone convolution layer (conv1), followed by 8 Fire modules (fire2-9) and a convolution layer(conv10), ending with a Softmax classification layer. It performs max-pooling

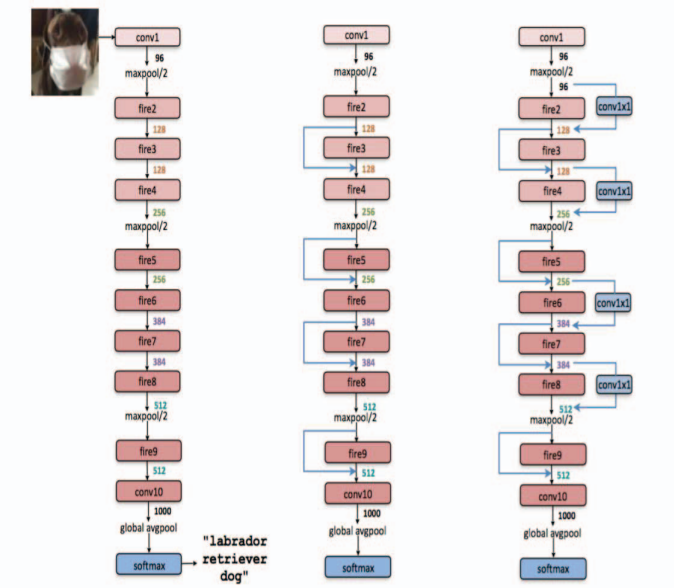

Fig. 2. Macro-architectural view of SqueezeNet architecture. Left: SqueezeNet ;Middle: SqueezeNet with simple bypass; Right: SqueezeNet with complex bypass.

with a stride of 2 after layers conv1, fire4, fire8, and conv10, and the activation function is ReLU. Its formula is as follows:

$$ReLU(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{1}$$

To avoid overfitting, we use dropout and extensive data enhancements. A dropout layer with rate = 0.5 after the fire9 module prevent co-adaptation between layers. The detection network resizes the input image to 227 x 227 x 3, runs a full convolutional network on the image to extract the feature vector of 1000 dimensions, and then sends it to Softmax for classification. Softmax is a generalization of logistic models on multi-class problemswhich has simple calculations and remarkable effects. Its formula is as follows:

$$S_i = \frac{e^{S_{yi}}}{\sum_{k=1}^{K} e^{S_k}} \tag{2}$$

S_i is the Softmax output corresponding to the correct category, and S_{y^i} is the linear score function corresponding to the correct category.

III. EXPERIMENT

In the first experiment, we use Raspberry Pi 3B and NCS2. Raspberry Pi 3B has Broadcom BCM2837 1.2GHZ processor, 1GB RAM and standard USB3.0 port. NCS2, with the Mygrid X VPU, can simultaneously process data from 12 cameras at a frame rate of 48FPS, which can reduce power consumption by

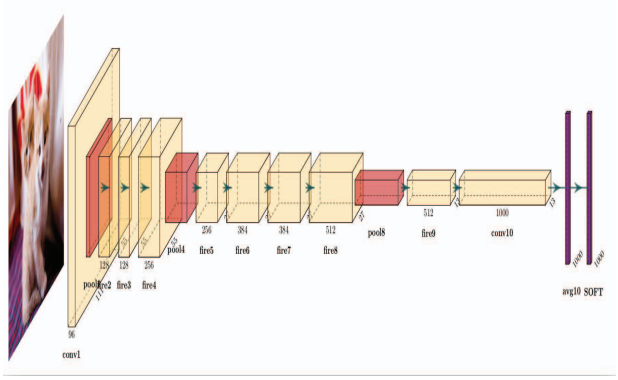

Fig. 3. The Architecture.

10 times compared to GPUs of the same performance. In the second experiment, we use the CPU, GPU and NCS2, where the CPU is Intel(R) Core(TM) i5-8500 CPU @ 3.00GHz, the GPU is GTX 1060ti.

A. Pi and Pi+NCS

We deploy the trained model on the embedded vision device Raspberry Pi 3B, and run it. The results of top5 on the input image are shown in Table 1.

TABLE I
CLASSIFICATION ACCURACY

Label	Pi	Pi+NCS
Siamese cat	0.975702	0.974121
Chihuahua	0.013853	0.014808
Egyptian cat	0.003637	0.003985
Space heater	0.001513	0.001662
Screen	0.000943	0.000764

Both of the Pi and the Pi with the NCS can accurately detect that the input image is a Siamese cat. Then, with the premise of guaranteeing accuracy, we compare the time of both, which is shown in Table 2.

TABLE II
DETECTION TIME

Time	Pi	Pi+NCS	Speedup
Time total	535.14(ms)	1156.12(ms)	**-2.16**×
Inference time	50.12(ms)	10.26(ms)	**4.88**×

The Time total contains the time to load the model and the detection category, and the Inference time only represents the time taken to detect the category. In terms of inference time, adding of the NCS can reach an acceleration of **4.88**× respectively.

In order to reduce the error, we increase the number of pictures detected until 300. The detection time of the top 5 pictures is shown in Table 3.

TABLE III
DETECTION TIME

Label	Pi	Pi+NCS	Speedup
Bike	535.14(ms)	1156.12(ms)	**-2.16**×
Cat	48.76(ms)	9.33(ms)	**5.23**×
Person	50.12(ms)	10.38	**4.83**×
Eskimo	50.78(ms)	10.28(ms)	**4.94**×
Chair	49.98(ms)	9.41(ms)	**5.31**×

The Pi takes 535.14ms in the first picture, and the Pi with NCS takes 1156.12ms. In the second picture, the Pi takes 48.76ms and the Pi with NCS takes 9.33ms. As the number of images increases, the Pi is stable at 50ms, and the Pi with NCS is stable at 9ms.

From the Fig.4, We can see that the Pi with NCS achieves approximately **4.71**× acceleration when the number of pictures is up to 300. In conclusion, With the NCS, Raspberry Pi can greatly improve the efficiency of computing.

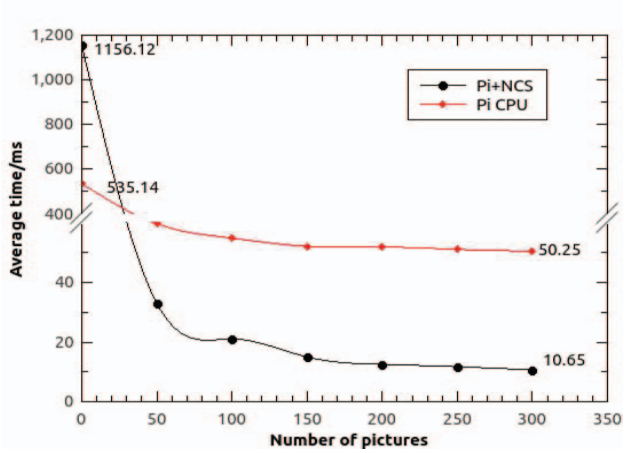

Fig. 4. The average time of different number of pictures between the Pi and the Pi with NCS

In order to prove the conclusion more correctly,we use SqueezeNet, GoogleNet, and AlexNet to test the Pi with or without the NCS. The results of comparison are shown in Table 4.

TABLE IV
RESULTS OF COMPARISON

Model	Pi	Pi+NCS	Speedup
SqueezeNet	50.25(ms)	10.65(ms)	**4.71**×
AlexNet	59.17(ms)	11.51(ms)	**5.14**×
GooleNet	88.07(ms)	13.64(ms)	**6.47**×

Table 4 shows that the Pi with NCS improves **4.71**× acceleration than the Pi in SqueezeNet, and **5.15**× acceleration in AlexNet, **6.47**× acceleration in GooleNet. As a result, the NCS can bring significant acceleration to the large neural network.

B. CPU GPU and NCS

In order to further demonstrate the high-speed computational performance of the NCS, we compare it directly with the CPU and GPU of the host. We run both vehicle type detection model and license plate number detection model in the same picture. The detection result of the NCS is shown in Fig.5.

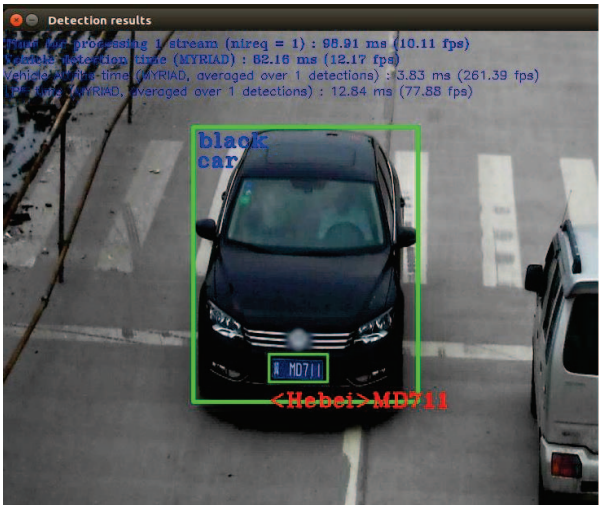

Fig. 5. Results of the NCS

The detection time of the three computing platforms is shown in the Table 5. Because it is not friendly to communicate with the PC via USB during NCS operation, it causes problems such as time delay when the image moves from the CPU to the NCS and returns the result. Therefore, it takes a long time to load the deep learning network model and the picture. The NCS is slower than the CPU in Time for processing 1 stream and Vehicle detection time. But when the picture is successfully loaded, the advantages of NCS are obvious. It can achieve **4.16**× and **2.05**× faster than the CPU in the Vehicle Attribute time and the License plate detection time, respectively. Especially, when the NCS is running, the CPU usage drops about by **20%**.

TABLE V
RESULTS OF COMPARISON

Time	CPU	GPU	NCS
Time for processing 1 stream	50.99(ms)	52.34(ms)	98.91(ms)
Vehicle detection time	21.35(ms)	18.35(ms)	82.16(ms)
Vehicle Attribute time	15.95(ms)	1.55(ms)	3.83(ms)
License plate detection time	26.36(ms)	2.35(ms)	12.84(ms)

The GPU processes the input image 2.47x faster than NCS in the Vehicle Attribute time and 5.46x faster in the License plate detection time. Although the calculation speed of the NCS is not as fast as the GPU, the NCS is lighter and cheaper than the GPU. Especially, it has lower power consumption than the GPU with the same performance.

IV. CONCLUSION

This paper analyses computing acceleration performance of the NCS compared to the CPU and GPU. This study proves that Raspberry Pi with the NCS has a **4 to 6** times improvement in computing performance compared to the Raspberry Pi alone. Compare to the CPU, the NCS can achieve **3**× acceleration. In addition, it can effectively drop-out the CPU usage. And compared to the GPU, it is lighter, cheaper and more energy efficient. In the era of data explosion in the future, it is more suitable for providing deep learning acceleration for mobile and embedded vision devices, and is more adaptable to future development trends. Our research results may provide guidance for the practical application of NCS. The NCS is a new accelerator for hardware, which has a significant development prospect.

ACKNOWLEDGMENT

I thank Jiayu Song, Jiangbo Ning, and Jianpin Yuan for helpful discussions and encouragement. This work is partially supported by the national natural science foundation of China(61601454).

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.
[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.
[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.
[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
[5] H. Derhamy, M. Andersson, J. Eliasson, and J. Delsing, "Workflow management for edge driven manufacturing systems," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*. IEEE, 2018, pp. 774–779.
[6] F. Jalali, O. J. Smith, T. Lynar, and F. Suits, "Cognitive iot gateways: automatic task sharing and switching between cloud and edge/fog computing," in *Proceedings of the SIGCOMM Posters and Demos*. ACM, 2017, pp. 121–123.
[7] V. Vujović and M. Maksimović, "Raspberry pi as a wireless sensor node: Performances and constraints," in *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2014, pp. 1013–1018.
[8] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, and K. Skadron, "A performance study of general-purpose applications on graphics processors using cuda," *Journal of parallel and distributed computing*, vol. 68, no. 10, pp. 1370–1380, 2008.
[9] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.