doi:10.3772/j. issn. 1006-6748. 2021. 01. 008

# LACC: a hardware and software co-design accelerator for deep neural networks①

Yu Yong (于　涌)②* ** ***, Zhi Tian*, Zhou Shengyuan***
(*State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, P. R. China)
(**School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, P. R. China)
(***Cambricon Technologies Ltd, Beijing 100190, P. R. China)

**Abstract**

With the increasing of data size and model size, deep neural networks (DNNs) show outstanding performance in many artificial intelligence (AI) applications. But the big model size makes it a challenge for high-performance and low-power running DNN on processors, such as central processing unit (CPU), graphics processing unit (GPU), and tensor processing unit (TPU). This paper proposes a LOGNN data representation of 8 bits and a hardware and software co-design deep neural network accelerator LACC to meet the challenge. LOGNN data representation replaces multiply operations to add and shift operations in running DNN. LACC accelerator achieves higher efficiency than the state-of-the-art DNN accelerators by domain specific arithmetic computing units. Finally, LACC speeds up the performance per watt by 1.5 times, compared to the state-of-the-art DNN accelerators on average.

**Key words**: deep neural network(DNN), domain specific accelerator, domain specific data type

## 0 Introduction

Nowadays, deep neural networks (DNNs) have become the dominating algorithms in many artificial intelligence (AI) applications, including image classification, object detection[1], natural language processing[2], speech recognition[3], image classification[4], and so on. However, deep neural networks are computation-intensive and memory-access-intensive tasks, due to the big model size and massive multiply accumulate (MAC) operations, which limit deep neural networks deployment. To accelerate processing of DNNs, many works on algorithms and hardware have been proposed.

As for algorithms aspect, there are two main methods to decrease models size and computation amount. Refs[5-8] applied model pruning to remarkably decrease model size. However, model pruning induces irregularity in networks, which is not hardware friendly. Refs[9-12] proposed quantization methods, to represent number with less bits compared to conventional deep neural networks.

As for hardware aspect, there are many novel domain-specific accelerators for neural networks[13-19]. Ref. [18] designed a neural networks accelerator family with sequential vector computation. Ref. [19] designed a spatial dataflow based neural network accelerator.

However, conventional processors cannot make full use of the benefit of the quantization algorithms and conventional quantization algorithms may lead to significant accuracy loss, which is intolerable in many AI application fields. This paper proposes LOGNN data representation and LACC accelerator with hardware and software co-design method to resolve the problem.

The key contributions are as follows.

A novel 8-bit width data representation (LOGNN) is proposed for deep neural networks. LOGNN data representation is a plug-and-play quantization method to reduce the model size of the neural networks and guarantee deep neural networks accuracy.

A novel processor LACC is proposed to take advantage of the LOGNN data representation by applying 3-bit integer operations during neural network processing.

① Supported by the National Key Research and Development Program of China (No. 2018AAA0103300, 2017YFA0700900, 2017YFA0700902, 2017YFA0700901, 2019AAA0103802, 2020AAA0103802).

② To whom correspondence should be addressed. E-mail: yuyong@ ict. ac. cn

Received on Jan. 5, 2020

A mapping method is introduced to deploy neural network layers of different sizes on LACC using LOGNN data representation.

## 1 LOGNN data representation

This section introduces the LOGNN data representation. Converting from floating-point data representation to LOGNN data representation only needs to find the nearest value in LOGNN data representation. The plug-and-play quantization method achieves quarter memory requirements of floating-point data type and replaces multiply operations with 3-bit integer operations. For a convolution layer, define $Y = N \times W$ as the output of the layer, where the input $N \in R^{c_{in} \times w_{in} \times h_{in}}$, weight $W \in R^{c_{out} \times c_{in} \times w \times h}$, the output $Y \in R^{c_{out} \times w \times h}$.

### 1.1 8-bit width data representation

Single-precision floating-point format are common data representation applied in deep neural networks. The single-precision floating-point format has 32 bits, which consist of exponent, mantissa, and sign[20], as shown in Fig. 1. The exponent is an 8-bit integer, the mantissa is a 23-bit integer, and the sign is a 1-bit integer.

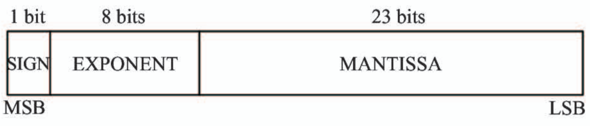

**Fig. 1** The components of floating-point data

When the number's exponent is equal to $2^8 - 1$ and the number's mantissa is not equal to 0, the number is NAN (not a number).

When the number's exponent is equal to $2^8 - 1$ and the number's mantissa is equal to 0, the number is INF (infinite).

When the number's exponent is not equal to 0 and less than $2^8 - 1$, the number is

$$(-1)^{SIGN} \times 2^{EXPONENT-15} \times (1 + MANTISSA \times 2^{-23}) \tag{1}$$

When the number's exponent is equal to 0 and less than $2^8 - 1$, the number is

$$(-1)^{SIGN} \times 2^{EXPONENT-15} \times MANTISSA \times 2^{-23} \tag{2}$$

Floating-point data format of real numbers supports a trade-off between range and precision. Exponent in 8 bits ranges from $2^{-127}$ to $2^{127}$. Mantissa in 23 bits ranges from 0 to $2^{23} - 1$.

Inspired by the conventional floating-point data representation, it proposes a new 8-bit floating-point data format (LOGNN) which is organized as in Fig. 2.

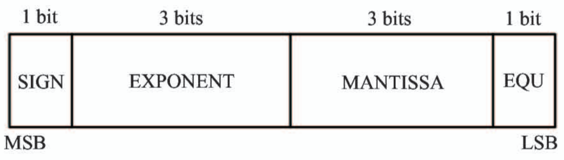

**Fig. 2** The components of the low bit width data representation

The 8-bit floating-point data format consists of 1-bit sign, 3-bit exponent, 3-bit mantissa, and 1-bit EQU. The sign bit represents the positive or negative of the number. Exponent is 3-bit-width ranging from 0 to 7. Mantissa ranges from 0 to $2^3 - 1$. EQU is 0 or 1, which enhances the precision and range of the data representation. The definition of LOGNN is the followings.

When the number's exponent is equal to $2^3 - 1$ and the number's mantissa is not equal to 0, the number is NAN.

When the number's exponent is equal to $2^3 - 1$ and the number's mantissa is equal to 0, the number is INF.

When the number's exponent is not equal to 0 and less than $2^3 - 1$, the number in normal mode is

$$(-1)^{SIGN} \times 2^{EXPONENT-3} \times (1 + MANTISSA \times 2^{-3} \times (1 + 2^{-EQU})) \tag{3}$$

When the number's exponent is equal to 0, the number in denormal mode is

$$(-1)^{SIGN} \times 2^{EXPONENT-3} \times MANTISSA \times 2^{-3} \times (1 + 2^{-EQU}) \tag{4}$$

The value of LOGNN in the highest precision is $2^{-3} \times 1 \times 2^{-3} \times (1 + 2^{-1})$, which is in denormal mode. The range of LOGNN is from $-1 \times 2^4 \times (1 + 7 \times 2^{-3} \times (1 + 2^0))$ to $1 \times 2^4 \times (1 + 7 \times 2^{-3} \times (1 + 2^0))$.

### 1.2 8-bit width data multiplication

The polynomial expansion of the proposed data representation in normal mode is rewritten as

$$(-1)^{SIGN} \times (2^{EXPONENT-3} + MANTISSA \times 2^{EXPONENT-6} + MANTISSA \times 2^{EXPONENT-EQU-6}) \tag{5}$$

The multiplication of two numbers in LOGNN format is

$$\begin{aligned} A \times B = (-1)^{SIGN_A xor SIGN_B} \times 2^{E_A + E_B - 6} \times (1 + 2^{-3} \times (M_A \\ + M_B + M_A \times M_B + (M_A + M_A \times M_B) \times 2^{-EQU_A} \\ + (M_B + M_A \times M_B) \times 2^{-EQU_B} \\ + M_A \times M_B \times 2^{-EQU_A - EQU_B})) \end{aligned} \tag{6}$$

Since $SIGN$, $E$(EXPONENT), $M$(MANTISSA) and $EQU$ all occupy less than 3 bits, only integer operation within 3 bits is needed, including one multiply operation ($M_A \times M_B$), four 3-bit integer add operations ($M_A + (M_A \times M_B)$, $M_B + (M_A \times M_B)$, $E_A + E_B$, $M_A + M_B$), one 1-bit integer add operation ($-EQU_A$

$- EQU_B$), and shift operations.

### 1.3 8-bit width data accumulation

Multiple accumulate operations can lead to overflow or underflow. The accumulate operations are common in deep neural networks, which is sensitive to data representation precision because of the huge numbers of parameters. LOGNN uses 32-bit integer for the middle results.

## 2 Experiments on LOGNN data representation

The proposed 8-bit LOGNN data representation is implemented in DNN inference without retraining conventional deep neural networks to verify the accuracy of LOGNN quantization method in networks inference.

LOGNN uses scale and zero-point to match floating-point values to the low bit width data representation with plug-and-play quantization method[21].

In this experiments, PyTorch[19] is used as programming framework and data format is modified in computation procedure. Both synapses and neurons use 8-bit LOGNN data format.

Table 1 shows the result of LOGNN data representation in AlexNet[20], ResNet18[21], and MobileNetV2[22]. LOGNN data representation decreases date width from 32 bits to 8 bits without accuracy loss. LOGNN approach also replaces the conventional multiply operation with 3-bit integer operations. In summary, the LOGNN data representation reduces 4 times storage and over 8 times computing complexity, compared to single-precision floating point data format.

Table 1 LOGNN data representation different deep neural networks

| Models | Bit width | Data type | Top-1 error |
|---|---|---|---|
| AlexNet Ref | 32 | FLOAT | 43.45% |
| AlexNet | 8 | LOGNN | 43.51% |
| ResNet18 Ref | 32 | FLOAT | 30.24% |
| ResNet18 | 8 | LOGNN | 30.28% |
| MobileNetV2 Ref | 32 | FLOAT | 27.81% |
| MobileNetV2 | 8 | LOGNN | 27.99% |

## 3 LACC deep neural networks accelerator

### 3.1 Domain specific arithmetic unit

There are many dedicated processors with domain specific architecture in deep neural networks. Hardware and software co-design in deep neural networks makes it conventional to design domain specific architecture to meet software requirements. LACC designs domain specific arithmetic unit for LOGNN data representation, which makes deep neural networks inference high efficiency.

In DNN forward and backward procedures, multiplication and accumulation are the most common operations:

$$\sum A \times B \tag{7}$$

With the LOGNN data representation, only 3-bit integer operations and accumulation operations are required. To maintain the accuracy in accumulation phase, LOGNN uses 32-bit integer for middle results with shift position. Eq. (6) is equal to Eq. (8).

$$\begin{aligned}(-1)^{SIGN_A xor SIGN_B} \times (2^{E_A+E_B-6} &+ (M_A + M_B + M_A \times M_B) \\ \times 2^{E_A+E_B-9} &+ ((M_A + M_A \times M_B) \times 2^{E_A+E_B-9-EQU_A} \\ &+ (M_B + M_A \times M_B) \times 2^{E_A+E_B-9-EQU_B} \\ &+ M_A \times M_B \times 2^{E_A+E_B-9-EQU_A-EQU_B}))\end{aligned} \tag{8}$$

This work designs domain specific arithmetic unit to support multiply and accumulation in LOGNN data representation. As shown in Eq. (8), LOGNN separates computations into two parts including exponential parameter computation and multiplier parameter computation.

Exponential parameter computation is shown in Fig. 3, which consists of three 3-bit integer add digital circuits and three 1-bit integer add digital circuit.

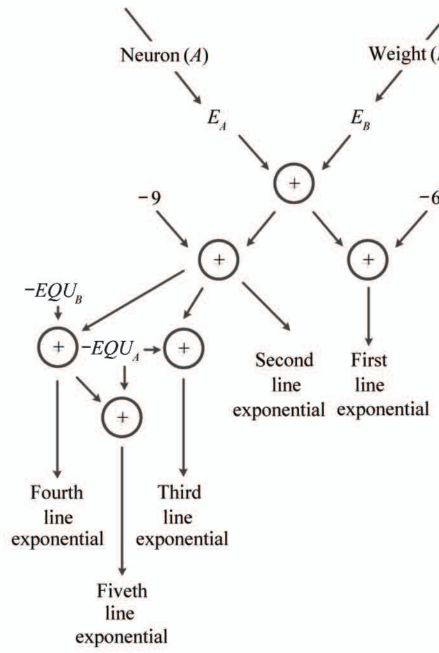

**Fig. 3** Exponential parameter computation

Multiplier parameter computation is shown in Fig. 4, which consists of one 3-bit integer multiply digital circuit and three 3-bit integer add digital circuit. Exponential parameters are used as the location in 32 bits integer on multiplier parameters.

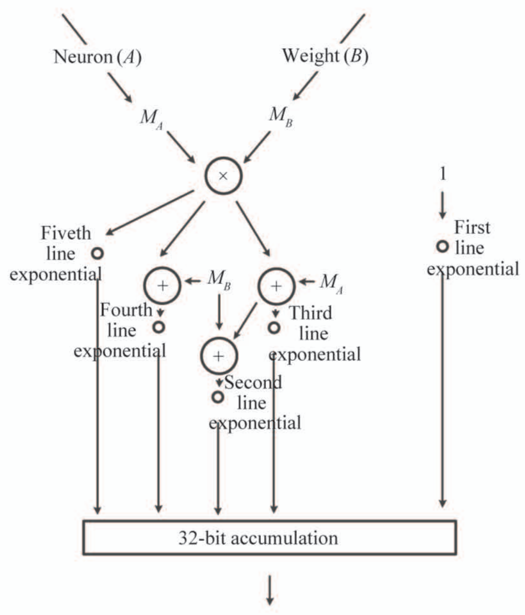

**Fig. 4** Multiplier parameter computation

Both of exponent parameter computation and multiplier parameter computation make up domain specific arithmetic unit. 32-bit accumulation is used in not only middle results of LOGNN but also middle neuron results of multiply and accumulation in neural networks for high accuracy arithmetic computation.

### 3.2 LACC deep neural networks accelerator

LACC is the deep neural networks accelerator to support LOGNN data representation and reuse data as much as possible. Fig. 5 shows the architecture of LACC.

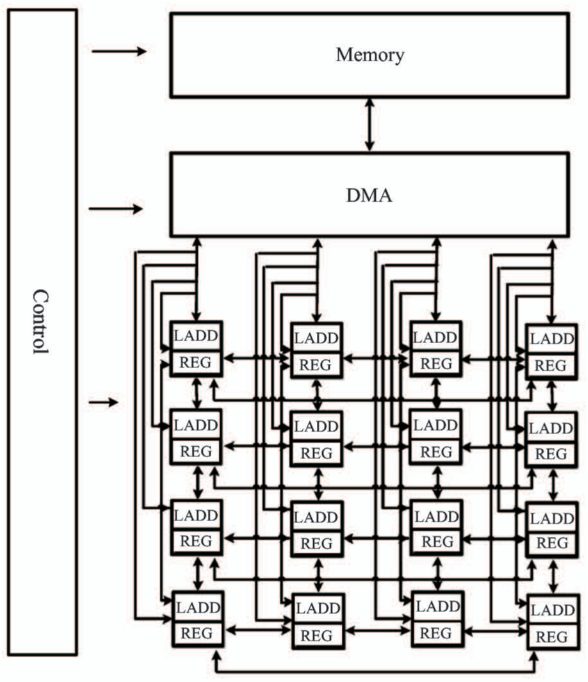

**Fig. 5** LACC: deep neural networks accelerator for LOGNN

LACC has no multiplier circuits. The key components of LACC are four units, including LADD matrix unit, control unit, memory unit and direct memory access (DMA) unit.

Control unit fetches an instruction of LACC and sends the control signals to the memory unit, DMA unit and LADD matrix unit. Memory unit is the storage module of LACC. It receives DMA's data fetching request and sends data to DMA module. DMA unit is the bridge between storage module and computing module of LACC. DMA receives micro instructions and transfers data as master. LADD matrix is the computing module of LACC, which receives control unit's control and DMA unit's data. LADD matrix unit is the computational component of LACC deep neural networks accelerator to finish multiply and accumulate operations. LADD matrix consists of LADD arithmetic unit, REG and ALU unit, and the network on chip linking them.

As mentioned before, LADD achieves multiply and accumulate operation on LOGNN data type with high performance and low power, which is the most computation and storage sensitivity of deep neural networks.

REG and ALU unit is the local storage unit near computing module and active computing module. REG and ALU unit are the base components of LADD matrix.

NOC links every computing modules as shown in Fig. 5. Column and row have been looped individually, which makes data transferring between each computing modules flexible. Transferring data between computing modules also makes much data reuse by transferring data between neighbor computing modules.

## 4 Mapping

### 4.1 Mapping of 2D convolution

Fig. 6 shows how LACC computes 2D convolution. Input feature maps slide on LADD matrix meanwhile multiplying weight and accumulating middle results. The detailed process is as follows.

First, input feature map is loaded to LADD matrix unit while multiplying and accumulating weight 1.

Second, input feature maps slide left one pixel and make multiply and accumulate operation on weight 2.

Third, input feature maps slide left one pixel and make multiply and accumulate operation on weight 3.

Fourth, input feature maps slide up one pixel and make multiply and accumulate operation on weight 6.

From the fifth to the ninth, input feature maps make a Z-sliding. Finally, the LADD matrix has the output feature map of a 3 ×3 2D convolution.

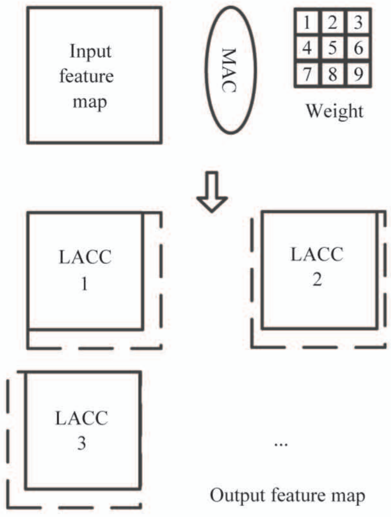

**Fig. 6** Convolution on LACC

### 4.2 Mapping of 3D convolution

Compared to 2D convolution, more 3D convolutions are used in deep neural networks. The 3D convolution has an additional channel dimension, and the neurons should be accumulated along this dimension. Both convolution layers and fully connected layers have the channel dimensions.

LACC's LADD matrix suits for accumulating the channel dimension. Once the 2D convolution is finished, DMA unit loads next input neurons of next feature map in channel dimension of the same output neurons. Then LADD matrix accumulates middle results of the channel dimension for the final 3D convolution results.

## 5 Experiments

### 5.1 Experiment methodology

The design of LACC is synthesized by Synopsys design compiler with TSMC 16 nm GP process in standard VT. The synthesized design is also placed and routed with the Synopsys ICC compiler. The design is simulated and verified with the Synopsys VCS, and estimated the power consumption with Synopsys PrimeTame PX based on the simulated value change dump (VCD) file. And taking AlexNet[22] as the benchmark network, the paper compares the LACC with three state-of-the-art hardware accelerators, including Eyeriss[23], ShiDianNao[19], and TPU[24].

### 5.2 Experiment results

The comparison between LACC and the other state-of-the-art accelerators in power cost and efficiency is shown in Table 2.

Table 2 Comparison LACC with other accelerators

| Accelerators | Alexnet Perf/fps | Power/mW | Efficiency/GOPs/s/W | Freq/MHz | Area/mm$^2$ | Technology |
|---|---|---|---|---|---|---|
| Eyeriss | 35 | 278 | 246 | 250 | 16 | TSMC65 |
| ShiDianNao | | 320 | 400 | 1000 | 5 | TSMC65 |
| TPU | | 40 000 | 2300 | 700 | <331 | TSMC28 |
| LACC | 131 682 | 66 | 7758 | 1000 | 1.6 | TSMC16 |

LACC is a scalable accelerator which can accommodate different sizes of LADD matrix. With 16 × 16 LADD matrix, the LACC accelerator in the experiment achieves a 256 MACs per cycle, which means 512 GOPs/s at 1 GHz. If not considering circuit area, the bigger LADD matrix, the higher performance can be achieved.

For fair comparison, Table 3 scales the baseline accelerators into TSMC 65 nm technology node[25]. TSMC 16 nm technology node is around 2.7 times efficiency than TSMC 28 nm and 6 times efficiency than TSMC 45 nm technology node, and 9 times efficiency than TSMC 65 nm technology node in AI application.

Therefore, Table 3 shows comparison of LACC in equivalent efficiency. LACC achieves 1.5 times efficiency as the other accelerators.

Table 3 Comparison of LACC in equivalent efficiency

| Accelerators | Equivalent Efficiency /TOPs/s/W | Freq/MHz | Technology |
|---|---|---|---|
| Eyeriss | 2.214 | 250 | TSMC16 |
| ShiDianNao | 3.6 | 1000 | TSMC16 |
| TPU | 5.3 | 700 | TSMC16 |
| LACC | 7.8 | 1000 | TSMC16 |

## 6 Conclusions

To resolve the intensive memory access and intensive computation problems of deep neural networks, this work proposes LOGNN data format and LACC hardware software co-design deep neural networks accelerator and removes the multiply operation in deep neural networks. LOGNN data format reduces the data width from 32 bits to 8 bits, and correspondingly reduces 4 times storage and memory bandwidth. LACC accelerator proposes domain specific arithmetic unit to suit LOGNN data representation while removing multiply operation. LADD matrix mapping of convolution makes middle result reuse as much as possible while processing multiply and accumulate operation. According to the experimental results, LACC deep neural networks accelerator achieves 1.5 times efficiency as the state-of-the-art deep neural networks.

## References

[ 1] Ren S, He K, Girshick R, et al. Faster R-CNN: towards real-time object detection with region proposal networks [C] // Advances in Neural Information Processing Systems, Montreal, Canada, 2017: 1137-1149

[ 2] Devlin J, Chang M W, Lee K, et al. Bert: pre-training of deep bidirectional transformers for language understanding[J]. *arXiv*:1810.04805, 2018

[ 3] Amodei D, Anubhai R, Battenberg E, et al. Deep speech 2: end-to-end speech recognition in English and Mandarin [J]. *Computer Science*, 2015(48),173-182

[ 4] Howard A, Sandler M, Chu G, et al. Searching for MobileNetV3[J]. *arXiv*:1905.02244, 2019

[ 5] Guo Y, Yao A, Chen Y. Dynamic network surgery for efficient DNNS[C] // Advances in Neural Information Processing Systems, Barcelona, Spain, 2016: 1387-1395

[ 6] Wang Y H, Xu C, You S, et al. CNNpack: packing convolutional neural networks in the frequency domain[C] // Advances in Neural Information Processing Systems, Barcelona, Spain, 2016: 253-261

[ 7] Han S, Mao H, Dally W J. Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding[J]. *Fiber*, 2015, 56 (4): 3-7

[ 8] Song H, Pool J, Tran J, et al. Learning both weights and connections for efficient neural network[C] // International Conference on Neural Information Processing Systems, Montreal, Canada, 2015: 1135-1143

[ 9] Gupta S, Agrawal A, Gopalakrishnan K, et al. Deep learning with limited numerical precision[C] // International Conference on Machine Learning, Lille, France, 2015: 1737-1746

[10] Rastegari M, Ordonez V, Redmon J, et al. Xnor-Net: imagenet classification using binary convolutional neural networks[C] // European Conference on Computer Vision, Amsterdam, Netherlands, 2016: 525-542

[11] Courbariaux M, Bengio Y, David J P. Binaryconnect: training deep neural networks with binary weights during propagations[C] // Advances in Neural Information Processing Systems, Montreal, Canada, 2015: 3123-3131

[12] Köster U, Webb T, Wang X, et al. Flexpoint: an adaptive numerical format for efficient training of deep neural networks[C] // Advances in Neural Information Processing Systems, Long Beach, USA, 2017: 1740-1750

[13] Chen T, Chen Y, Duranton M, et al. BenchNN: on the broad potential application scope of hardware neural network accelerators [C] // The 2012 IEEE International Symposium on Workload Characterization, La Jolla, USA, 2012: 36-45

[14] Du Z, Palem K, Lingamneni A, et al. Leveraging the error resilience of machine-learning applications for designing highly energy efficient accelerators[C] // The Asia and South Pacific Design Automation Conference, Singapore, 2014: 201-206

[15] Zhang S, Du Z, Zhang L, et al. Cambricon-x: an accelerator for sparse neural networks[C] // The 49th Annual IEEE/ACM International Symposium on Microarchitecture, Taibei, China, 2016: 1-12

[16] Liu S, Du Z, Tao J, et al. Cambricon: an instruction set architecture for neural networks[C] // The 43rd International Symposium on Computer Architecture, Seoul, Korea, 2016: 393-405

[17] Chen Y, Luo T, Liu S, et al. DaDianNao: a machine-learning supercomputer[C] // The Annual International Symposium on Microarchitecture, Cambridge, UK, 2015: 609-622

[18] Chen T, Du Z, Sun N, et al. DianNao: a small-footprint high-throughput accelerator for ubiquitous machine-learning[C] // The 19th International Conference on Architectural Support for Programming Languages and Operating Systems, Salt Lake City, USA, 2014: 269-284

[19] Du Z, Fasthuber R, Chen T, et al. ShiDianNao: shifting vision processing closer to the sensor[C] // The 42nd Annual International Symposium on Computer Architecture, Portland, USA, 2015: 92-104

[20] IEEE Computer Society. 754-2008 IEEE standard for floating-point arithmetic [S]. New York, USA: IEEE Computer Society Standards, 2008

[21] Jacob B, Kligys S, Chen B, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference[C] // IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, 2018: 2704-2713

[22] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks[C] // Advances in Neural Information Processing Systems, Harrahs and Harveys, Lake Tahoe, USA, 2012: 1097-1105

[23] Chen Y H, Emer J, Sze V. Eyeriss: a spatial architecture for energy-efficient dataflow for convolutional neural networks[C] // ACM SIGARCH Computer Architecture News, Seoul, Korea, 2016: 367-79

[24] Jouppi N P, Young C, Patil N, et al. In-datacenter performance analysis of a tensor processing unit[C] // International Symposium on Computer Architecture, Toronto, Canada, 2017: 1-12

[25] Khazraee M, Zhang L, Vega L, et al. Moonwalk: NRE optimization in ASIC clouds[J]. *ACM SIGOPS Operating Systems Review*, 2017, 51(2): 511-26

**Yu Yong**, born in 1992. He received his B.S. degree in intelligence science and technology from Nankai University in 2015. He is currently a Ph.D candidate at Institute of Computing Technology, Chinese Academy of Sciences. His research interests include computer architecture and machine learning algorithms.