Efficient Optimization and Hardware Acceleration of CNNs towards the Design of a Scalable Neuro-inspired Architecture in Hardware

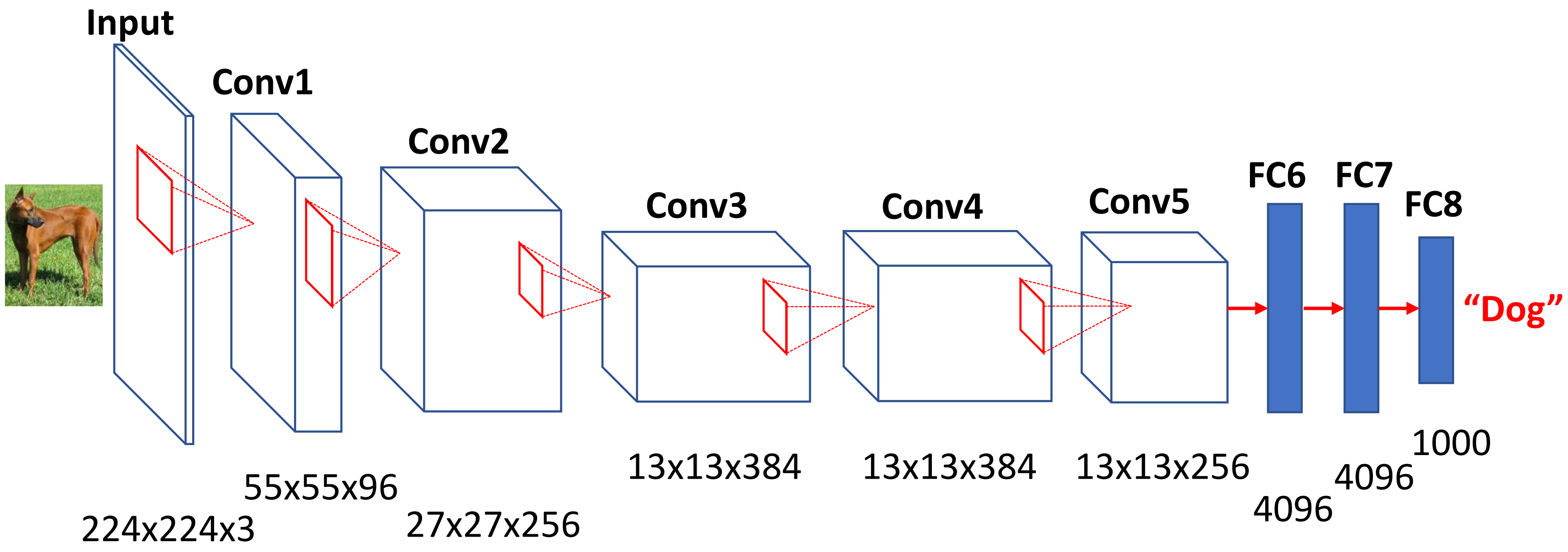
The H. Vu, Ryunosuke Murakami,
Yuichi Okuyama, Abderazek Ben Abdallah

University of Aizu
Graduate School of Computer Science and Engineering
Adaptive Systems Laboratory

BigCom2018, January 15-18, 2018, Shanghai, China

Convolution Neural Network

CNNs are attractive in computer vision tasks such as image classification.



[AlexNet-Krizhevsky'12]

- 5 ConvNets, 3 Fully-Connected, 1000 classes.
- Error rate of 15.3% in ImageNet Challenge.

Implementation Platforms for ANNs

	Analog ASIC	**Digital ASIC**	**FPGA**	**Processor Based**	**Parallel Computer**
Speed	+++	++	+	–	+
Area	+++	++	+	–	––
Cost	––	––	++	++	–
Design time	––	––	++	+++	+
Reliability	––	+	++	++	++

– –: very unfavourable, –: unfavourable, +: favourable,
++: very favourable, +++: highly favourable *[Omondi'06]*

→ Balancing

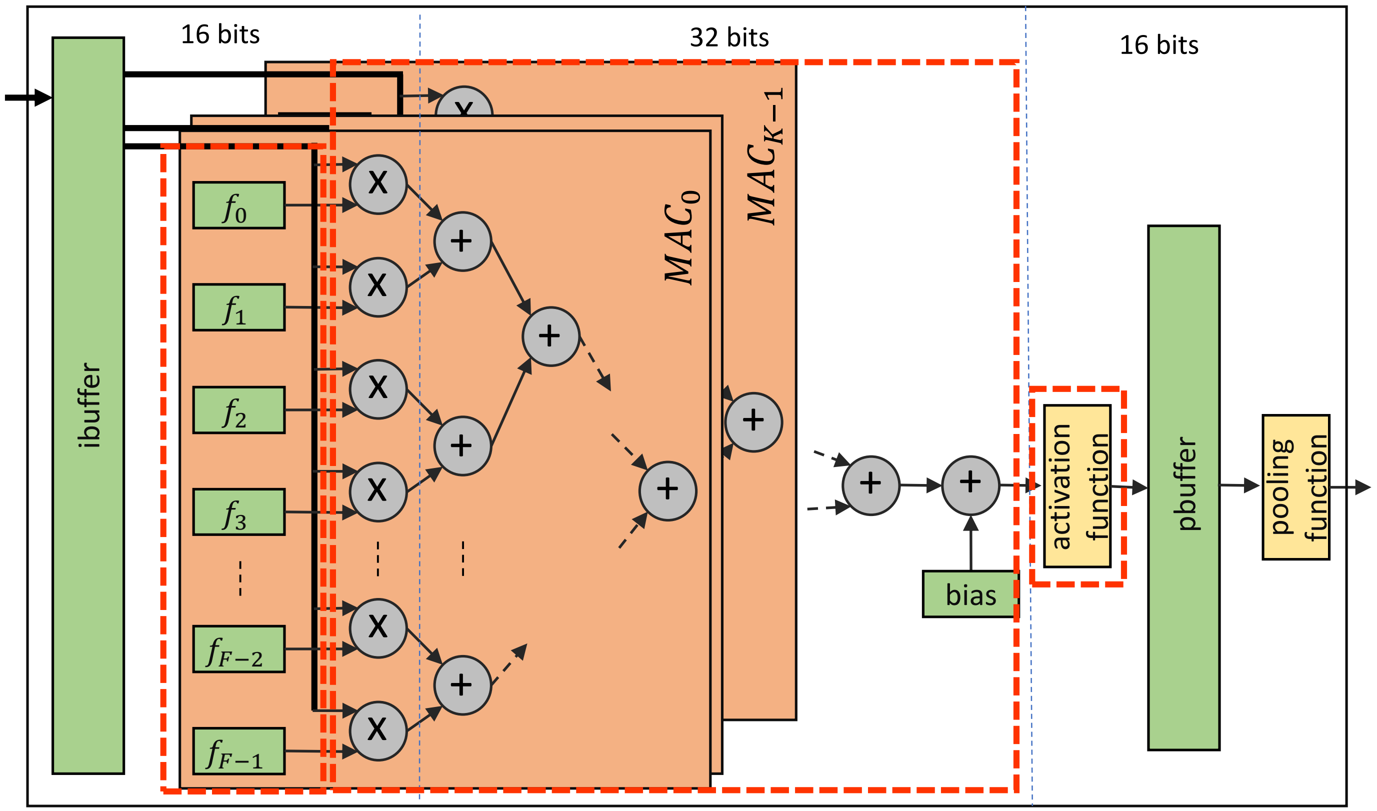
Related works

- *[Yongmei’15]* proposed an accelerator for image recognition based on CNN: Vertex 7 FPGA, 11-bits fixed-point precision. **Low accuracy** (96.8% for MNIST)
- *[Ghaffari’16]* presented two FPGA based accelerators for CNN using high level synthesize. **High latency** (2ms and 51ms for one MNIST example).
- *[Zhou’15]* introduced FPGA design for PCANet. This architecture achieved high accuracy (99.46% for MNIST) and short execution time $7.6\mu s/\text{example}$ but using a **significant amount of area cost**.

Paper contributions

- An FPGA-based CNN architecture has an efficient balance between hardware complexity, execution time, and accuracy:
  - Based on a pipeline processing unit with three layers of neural processing.
  - Use high accuracy approximation for implementing activation function.
  - Use off-chip learning to reduce hardware complexity and take full advantage of the software calculation precision.

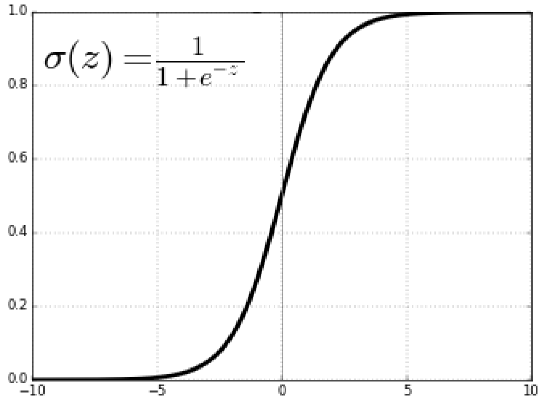
Neural processing unit



- Off-chip learning.
- Fixed point calculation and Rounding-to-nearest-integer method.
- Activation function approximation

Activation function Appoximation

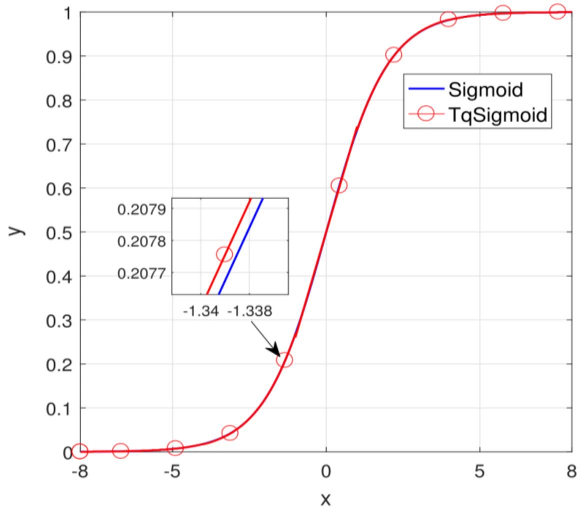
[Sigmoid activation function]



[Twenty-five piece-wise intervals for sigmoid function approximation]

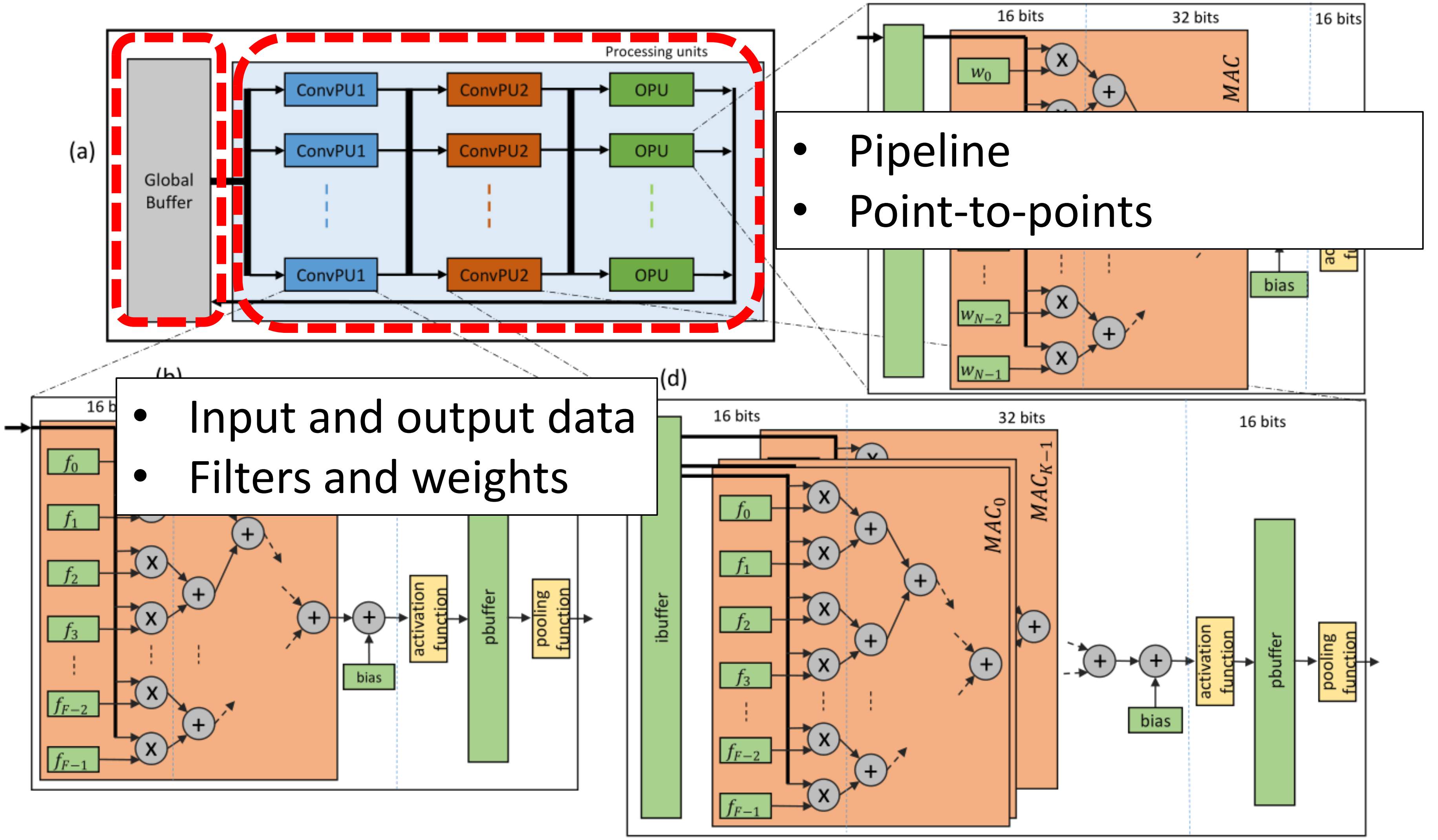
Interval	Function Form
(1,1)	$y = 0.2383x + 0.5000$
[-2,-1]	$y = 0.0467x^2 + 0.2896x + 0.5118$
[-3,-2)	$y = 0.0298x^2 + 0.2202x + 0.4400$
[-4,-3)	$y = 0.0135x^2 + 0.1239x + 0.2969$
[-5,-4)	$y = 0.0054x^2 + 0.0597x + 0.1703$
[-5.03,-5)	$y = 0.0066$
[-5.2,-5.03)	$y = 0.0060$
[-5.41,-5.2)	$y = 0.0050$
[-5.66,-5.41)	$y = 0.0400$
[-6,-5.66)	$y = 0.0030$
[-6.53,-6)	$y = 0.0020$
[-7.6,-6.53)	$y = 0.0010$
[-∞,-7.6)	$y = 0$
[1,2)	$y = -0.0467x^2 + 0.2896x + 0.4882$
[2,3)	$y = -0.0298x^2 + 0.2202x + 0.5600$
[3,4)	$y = -0.0135x^2 + 0.1239x + 0.7030$
[4,5)	$y = -0.0054x^2 + 0.0597x + 0.8297$
[5,5.0218)	$y = 0.9930$
[5.0218,5.1890)	$y = 0.9940$
[5.1890,5.3890)	$y = 0.9950$
[5.3890, 5.6380)	$y = 0.9960$
[5.6380,5.9700)	$y = 0.9970$
[5.9700,6.4700)	$y = 0.9980$
[6.4700,7.5500)	$y = 0.9990$
[7.5500, +∞)	$y = 1$

[Sigmoid approximation result]



$$MSE = 1.09 \times 10^{-6}$$

System Architecture



- Pipeline
- Point-to-points

- Input and output data
- Filters and weights

Evaluation methodology

- Application: handwritten digit recognition with MNIST dataset.
- Implementation:
  - *Hardware*: Verilog HDL, Vivado 2017.1, Virtex-7 v2000tflg1925-1.
  - *Software*: Matlab, 3.40 GHz Intel Core-i7 4770 and 16 GB of RAM.
  - Off-chip learning.
- Objectives:
  - Hardware complexity.
  - Performance.
  - Accuracy.

Result (1/3)

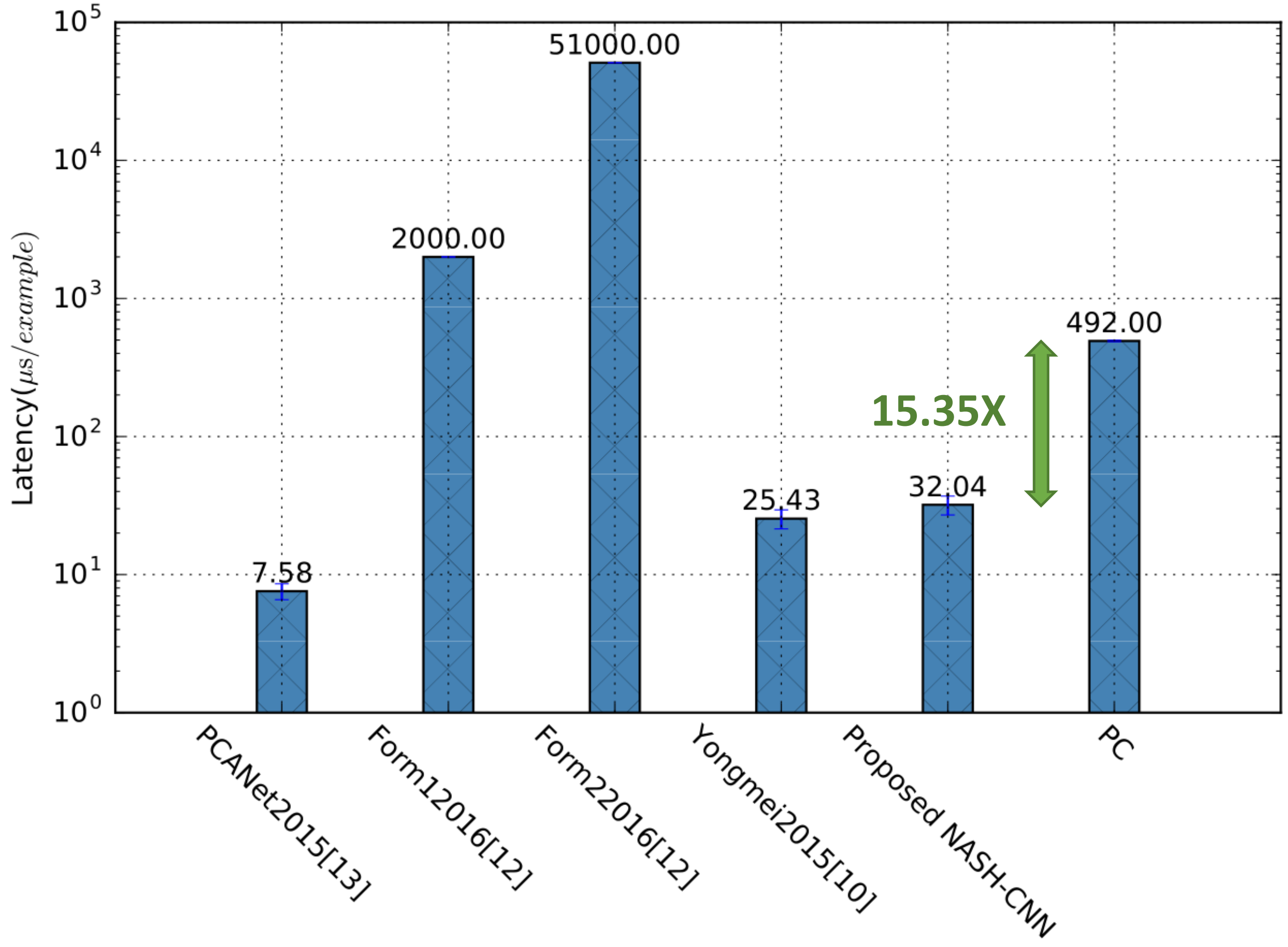
[Area cost]

Parameters/Systems	**PCANet'15**	**Form1'16**	**Form2'16**	**Yongmei'15**	**Proposed work**
Target device	Virtex-7 980T	Xilinx Zynq	Xilinx Zynq	Virtex-7 vx485tffg1761-2	Virtex-7 v2000tflg1925-1
FF/Register(F/R)	358848-R	54075-F	35399-F	66364-F	18299-F
LUT	265460	14832	39879	51125	65386
BRAM	64 (36E1)	27	3	0	0
DSP	3599 (48E1)	20 (48E)	90 (48E)	638 (48E	1095 (48E1)
Precision	-	25-bits fixed-point	25-bits fixed-point	11-bits fixed-point	16-bits fixed-point

- Smallest number of flip-flop
- Figures for proposed work are less than PCANet'15

Result (2/3)

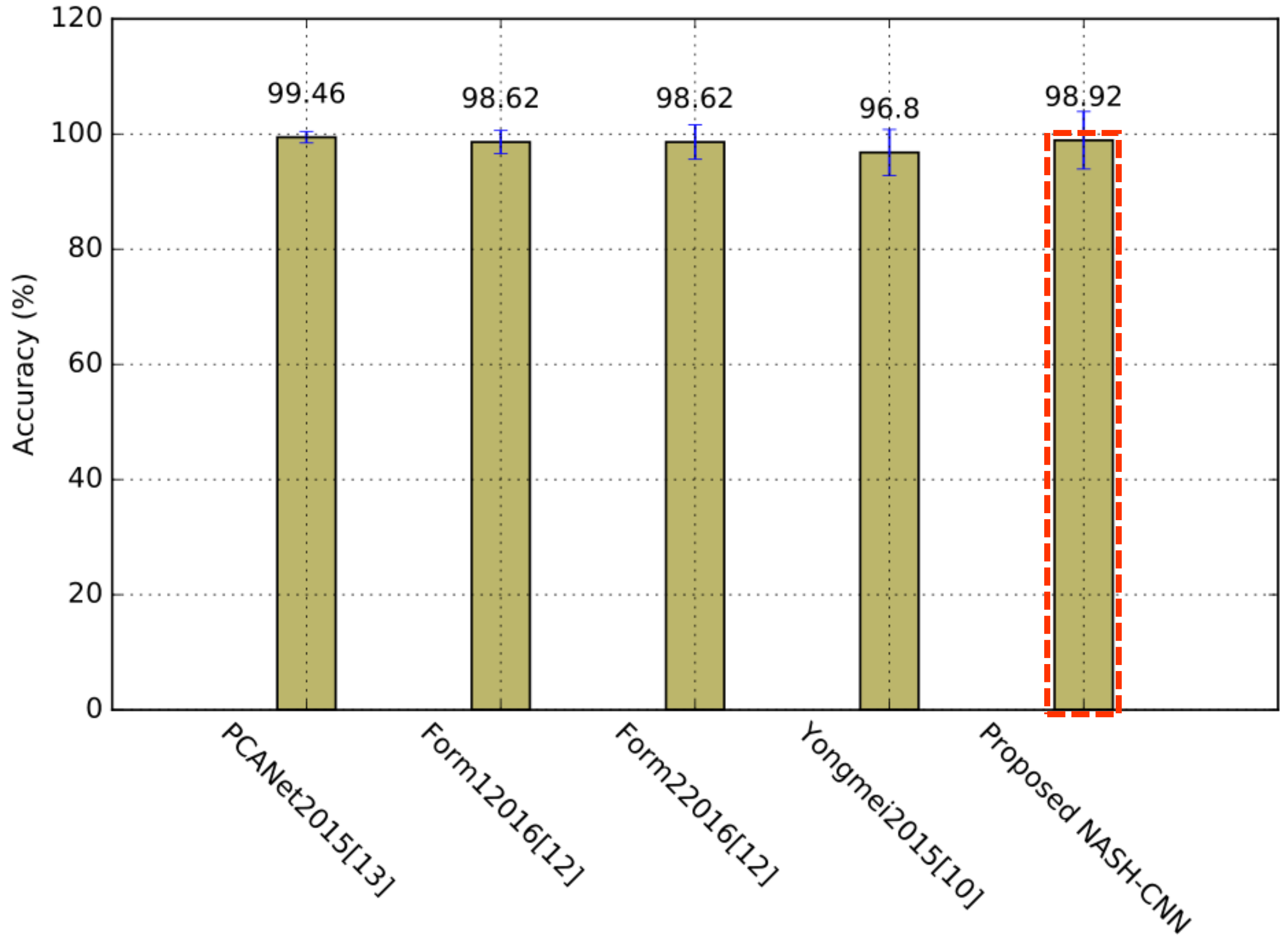
[Execution time]



- Speed up **15.35X** compared to software implementation

Result (3/3)

[Accuracy]



- Higher compared to the others excepting PCANet

Conclusion

- Architecture for hardware-based CNNs is proposed and implemented.
- Evaluation results show that the system achieves an efficient trade-off between area cost, accuracy, and latency.
- Furthermore, the current architecture is our first step towards the design of scalable spiking architecture in hardware based on our 3D Network-on-Chip architecture.

Thank you for your attention!

THE END