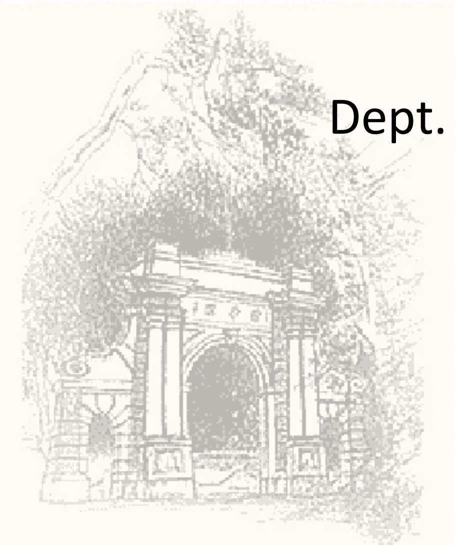
Binary Convolutional Neural Network on RRAM

Tianqi Tang, Lixue Xia, Boxun Li, Yu Wang, Huazhong Yang

Dept. of E.E, Tsinghua National Laboratory for Information Science and Technology (TNList)

Tsinghua University, Beijing, China

e-mail: yu-wang@mail.tsinghua.edu.cn



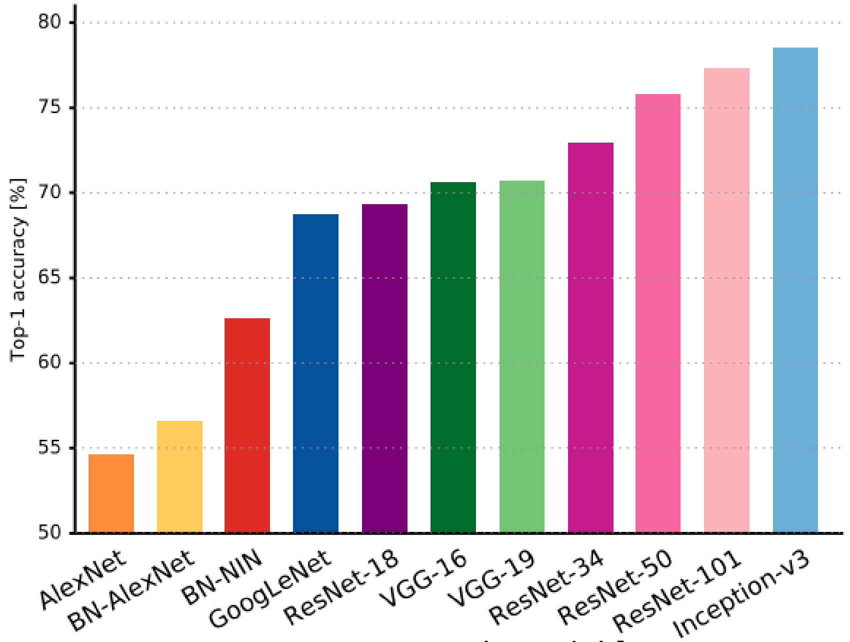
Outline

Deep Learning
Better Performance & Larger FLOPS

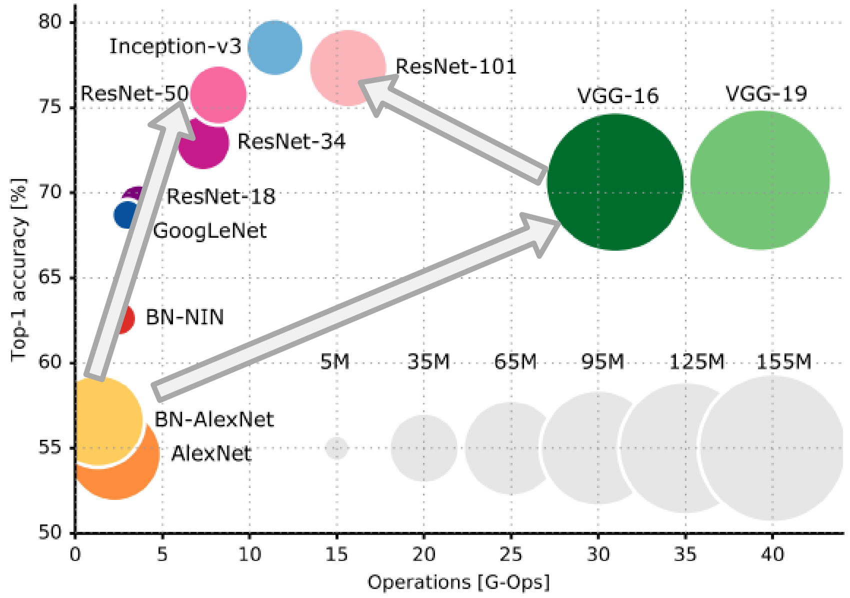
Winners of the Image-Net Large-Scale Visual Recognition Challenge (ILSVRC)

Task: Classification & Localization with Provided Data

	2011	2012	2013	2014	2015	2016
Team	XRCE	SuperVision	Clarifai	VGG	MSRA	Trimps-Soushen
Model	Not CNN	AlexNet	ZF	VGG-16	ResNet-152	Ensemble
Err (Top-5)	25.8%	16.4%	11.7%	7.4%	3.57%	2.99%



Top1 Accuracy v.s. Network Model [arxiv: 1605.07678]



Top1 Accuracy v.s. GOPs, Model Size [arxiv: 1605.07678]

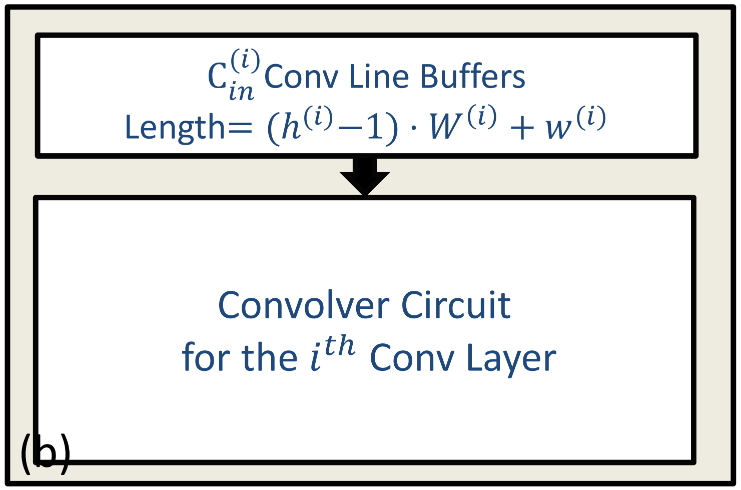
Deep Learning: Network Quantization

- Binary Neural Network
  - Extreme Quantization: binary weight, binary activation
  - Workflow: Quantization while Training

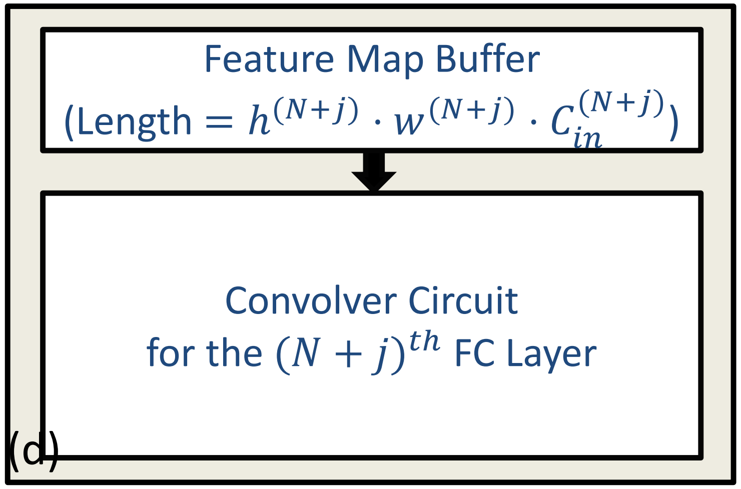
Forward	Backward	Update	Quantize
Integer	Integer	Float	Float2Int
$Y = W^i \cdot X$	$dX = dY \cdot W^i$ $dW^i = dx * dy$	$dW^f = Q' \cdot dW^i$ $W^f = W^f + dW^f$	$W^i = Q(W^f)$

BCNN Accelerator Design

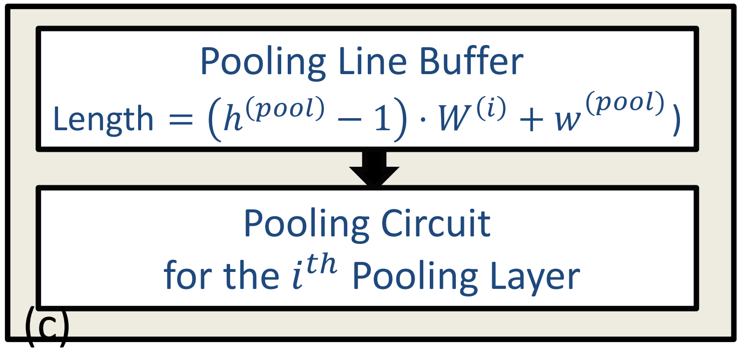
- Convolution Layer


$C_{in}^{(i)}$ Conv Line Buffers
Length= $(h^{(i)}-1)\cdot W^{(i)}+w^{(i)}$
Convolver Circuit
for the i^{th} Conv Layer
(b)

Fully-Connected Layer

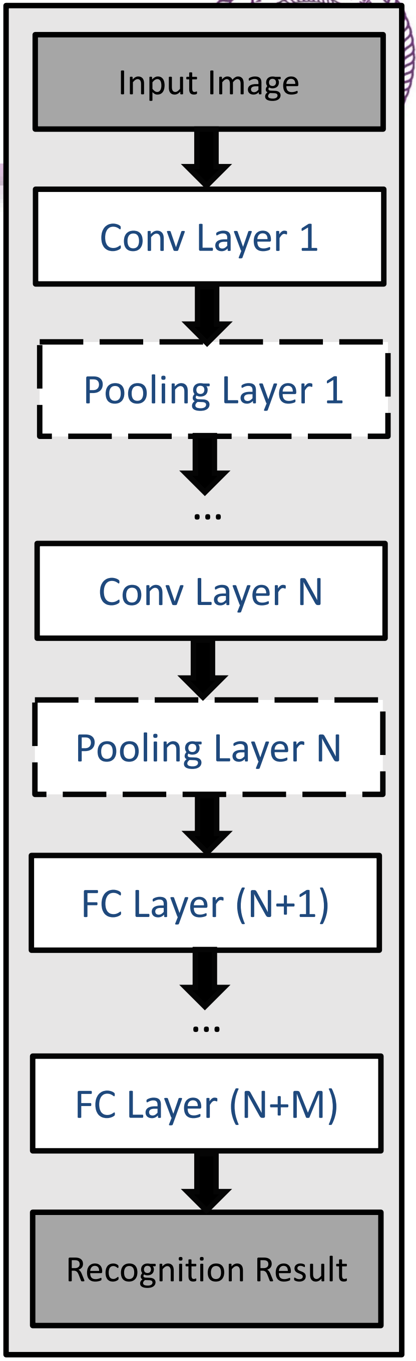

Feature Map Buffer
(Length $= h^{(N+j)}\cdot w^{(N+j)}\cdot C_{in}^{(N+j)}$)
Convolver Circuit
for the $(N+j)^{th}$ FC Layer
(d)

- Pooling Layer


Pooling Line Buffer
Length $= (h^{(pool)}-1)\cdot W^{(i)}+w^{(pool)}$)
Pooling Circuit
for the i^{th} Pooling Layer
(c)

(1) How to design the Convolver Circuit?

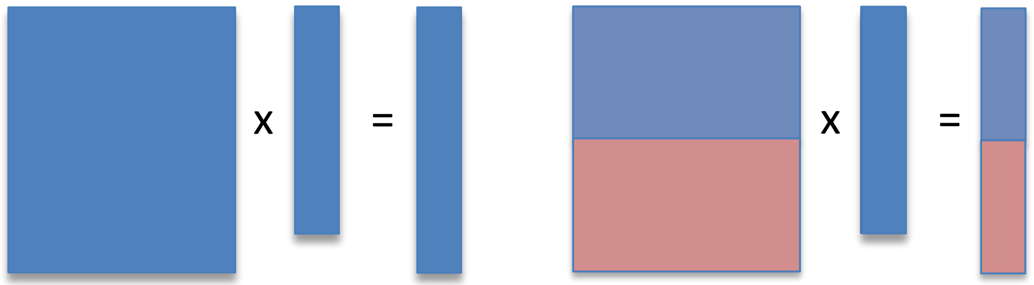
(2) How to design the Conv/Pooling Line Buffer?



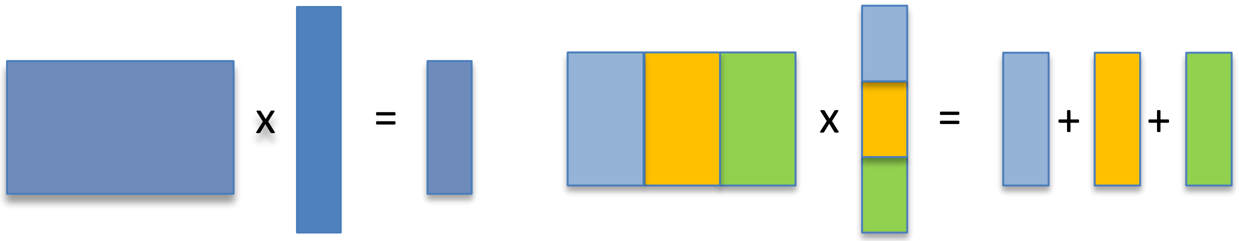
BCNN Accelerator Design: Convolver Circuit

How to map a large matrix onto a group of RRAM crossbars?

- Column Splitting

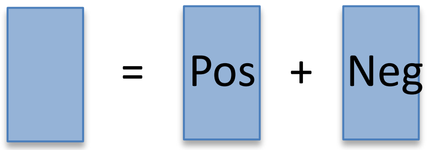


- Row Splitting



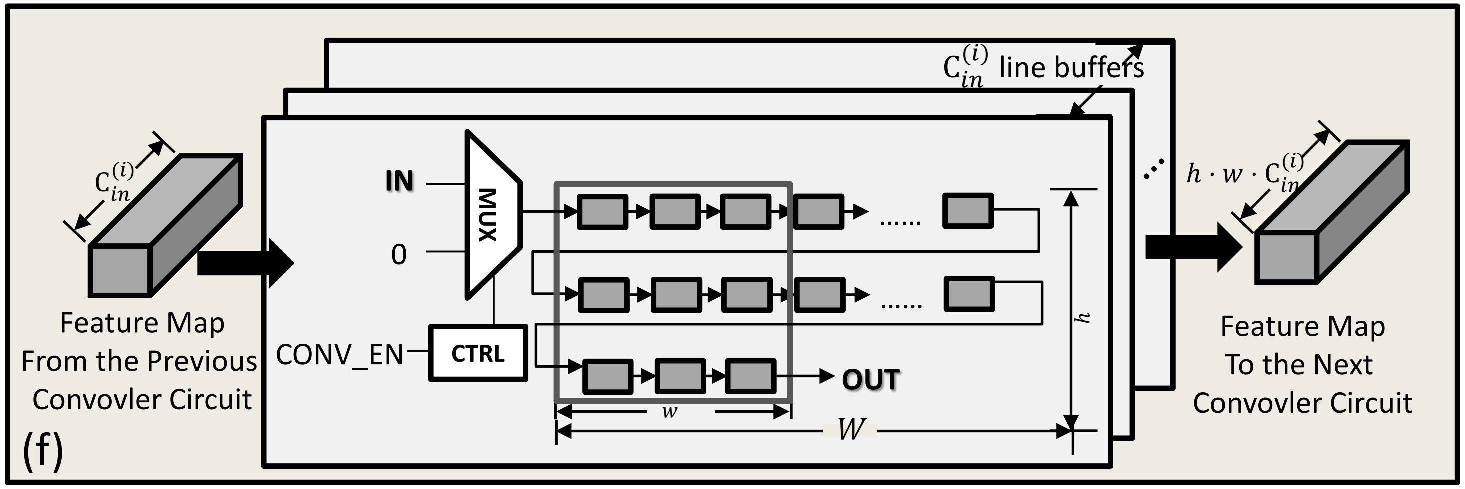
What is the bit length of the partial sum?

- Signal Splitting



BCNN Accelerator Design: Line Buffer & Pipeline

- Sliding Window
  - the Convolver circuit can **awake (A)** from **sleep (S)** once the input data of the Conv kernel size is achieved;
  - **Unnecessary** to buffer the whole input feature maps.
  - Structure of **Line Buffer** introduced.



$$T_{\text{pip}} = (W^{(1)}+p)\cdot(H^{(1)}+2p) + \sum_{i\in\text{Conv}}^{i>1}(W^{(i)}+p) + \sum_{i\in\text{Pool}} 1 + \sum_{j\in\text{FC}} 1$$

  - Much smaller than $T_{\text{straight\_forward}}$

BCNN Accelerator Design: Line Buffer & Pipeline

- Dataflow Example on VGG11
  - Conv-Conv

	T_0	T_1	T_2	…	T_W	T_{W+1}	T_{W+2}	T_{W+3}	
$Conv_K$ LB Input	$x_{2,1}^{(k)}$	$x_{2,2}^{(k)}$	$x_{2,3}^{(k)}$	…	$x_{2,W}^{(k)}$	0	$x_{3,1}^{(k)}$	$x_{3,2}^{(k)}$	…
$Conv_K$ Xbar	S	A	A	…	A	A	S	A	…
$Conv_{K+1}$ LB Input	0	$x_{1,1}^{(k+1)}$	$x_{1,2}^{(k+1)}$	…	$x_{1,W-1}^{(k+1)}$	$x_{1,W}^{(k+1)}$	0	$x_{2,1}^{(k+1)}$	…
$Conv_{K+1}$ Xbar	S	S	A	…	A	A	A	S	…

BCNN Accelerator Design: Line Buffer & Pipeline

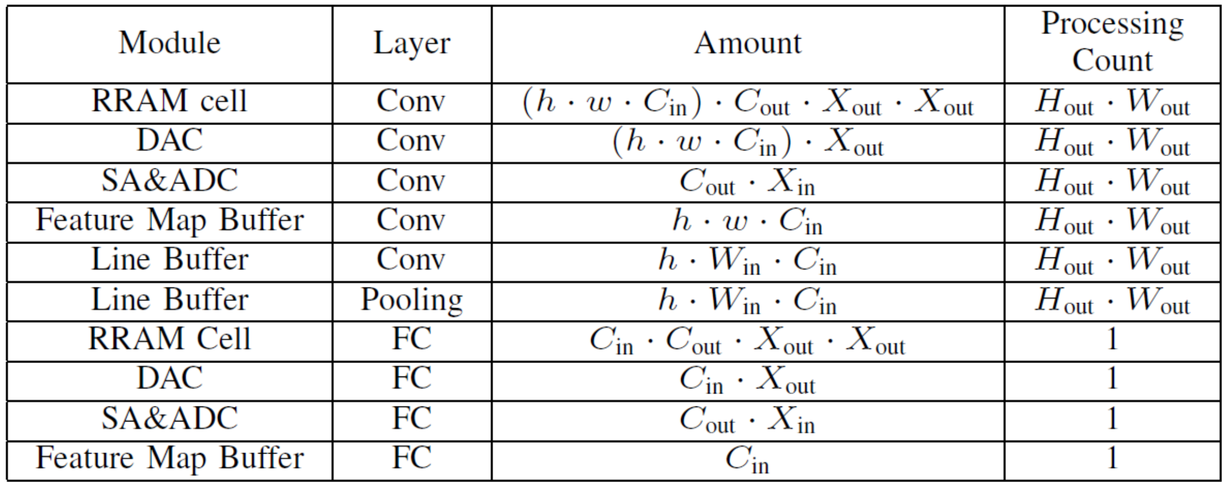
- Dataflow Example on VGG11
  - Conv-Pooling-Conv

	T_0	T_1	T_2			…	T_W	T_{W+1}	T_{W+2}	T_{W+3}	…	T_{2W+1}
$Conv_K$ LB Input	0	$x_{3,1}^{(k)}$	$x_{3,2}^{(k)}$	$x_{3,3}^{(k)}$	$x_{3,4}^{(k)}$	…	$x_{3,W}^{(k)}$	0	$x_{4,1}^{(k)}$	$x_{4,2}^{(k)}$	…	$x_{4,W}^{(k)}$
$Conv_K$ Xbar	A	S	A	A	A	…	A	A	S	A	…	A
$Pool_K$ LB Input	$y_{1,W}^{(k)}$	X	$y_{2,1}^{(k)}$	$y_{2,2}^{(k)}$	$y_{2,3}^{(k)}$	…	$y_{2,W-1}^{(k)}$	$y_{2,W}^{(k)}$	X	$y_{3,1}^{(k)}$	…	$y_{3,W-1}^{(k)}$
$Pool_K$ Circuit	A	S	S	A	S	…	S	A	S	Sleep until new data coming		
$Conv_{K+1}$ LB Input	0	0	0	$x_{1,1}^{(k+1)}$	0	…	0	$x_{1,\frac{W}{2}}^{(k+1)}$	0	0		

Experimental Results

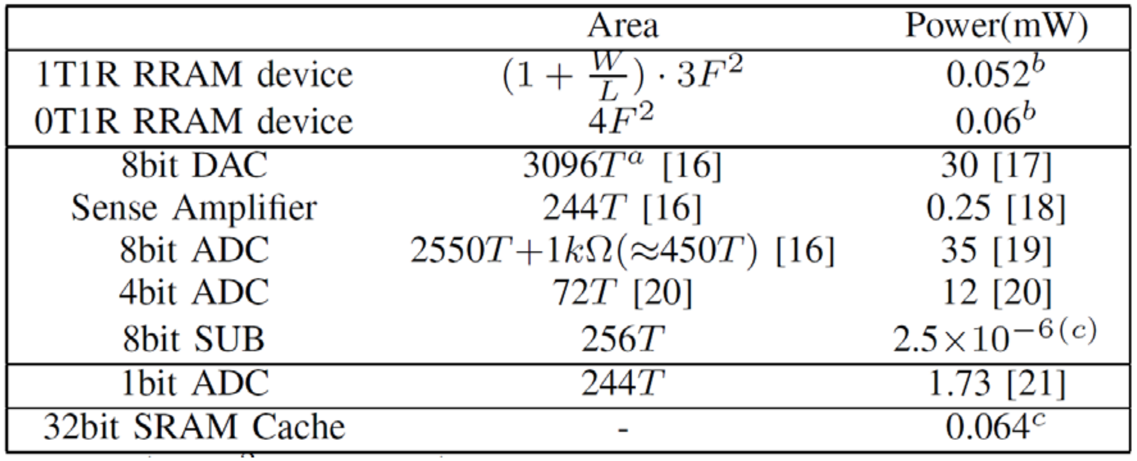
- SetUp:

TABLE II: Amount and Processing Count of Computing Units, Interfaces and Buffers



Module	Layer	Amount	Processing Count
RRAM cell	Conv	$(h \cdot w \cdot C_{\text{in}}) \cdot C_{\text{out}} \cdot X_{\text{out}} \cdot X_{\text{out}}$	$H_{\text{out}} \cdot W_{\text{out}}$
DAC	Conv	$(h \cdot w \cdot C_{\text{in}}) \cdot X_{\text{out}}$	$H_{\text{out}} \cdot W_{\text{out}}$
SA&ADC	Conv	$C_{\text{out}} \cdot X_{\text{in}}$	$H_{\text{out}} \cdot W_{\text{out}}$
Feature Map Buffer	Conv	$h \cdot w \cdot C_{\text{in}}$	$H_{\text{out}} \cdot W_{\text{out}}$
Line Buffer	Conv	$h \cdot W_{\text{in}} \cdot C_{\text{in}}$	$H_{\text{out}} \cdot W_{\text{out}}$
Line Buffer	Pooling	$h \cdot W_{\text{in}} \cdot C_{\text{in}}$	$H_{\text{out}} \cdot W_{\text{out}}$
RRAM Cell	FC	$C_{\text{in}} \cdot C_{\text{out}} \cdot X_{\text{out}} \cdot X_{\text{out}}$	1
DAC	FC	$C_{\text{in}} \cdot X_{\text{out}}$	1
SA&ADC	FC	$C_{\text{out}} \cdot X_{\text{in}}$	1
Feature Map Buffer	FC	C_{in}	1

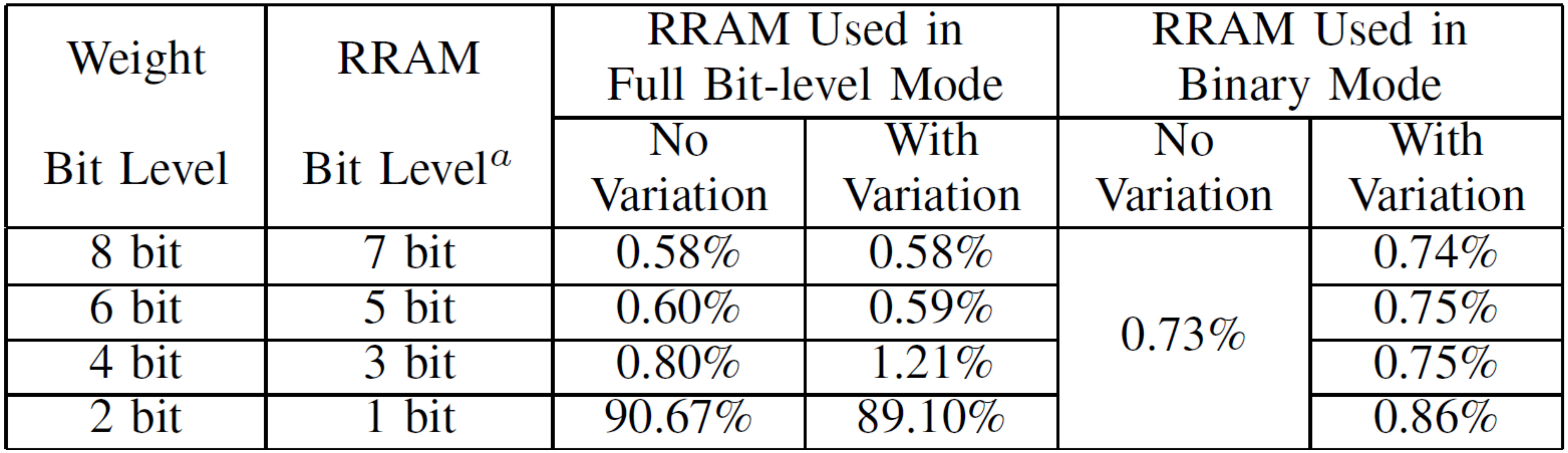
TABLE III: Area and Power Cost of Circuit Elements



	Area	Power(mW)
1T1R RRAM device	$(1+\frac{W}{L}) \cdot 3F^2$	0.052^b
0T1R RRAM device	$4F^2$	0.06^b
8bit DAC	$3096T^a$ [16]	30 [17]
Sense Amplifier	$244T$ [16]	0.25 [18]
8bit ADC	$2550T+1k\Omega(\approx 450T)$ [16]	35 [19]
4bit ADC	$72T$ [20]	12 [20]
8bit SUB	$256T$	$2.5\times10^{-6(c)}$
1bit ADC	$244T$	1.73 [21]
32bit SRAM Cache	-	0.064^c

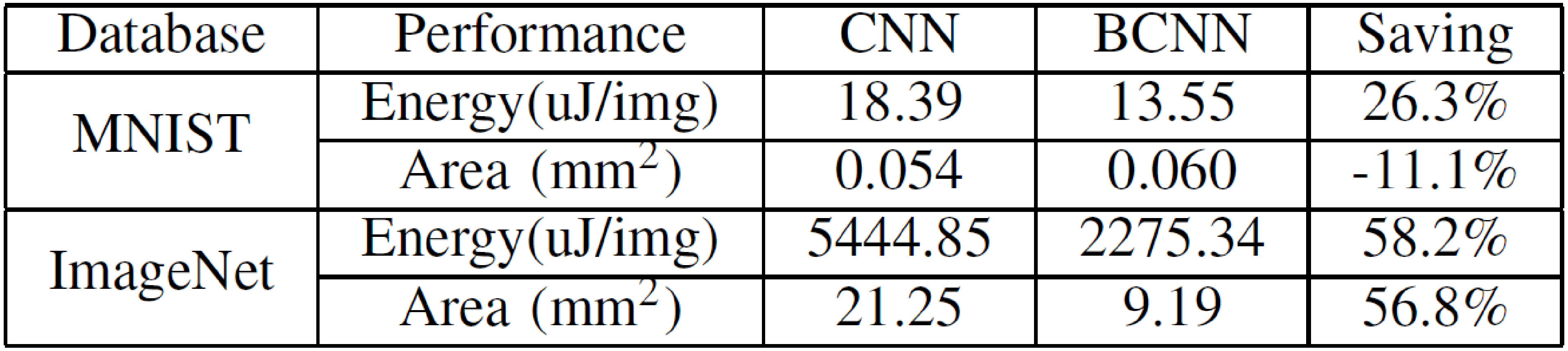
Experimental Results

- Accuracy: Effects of Device Variation Under Different Bit-Levels



Weight Bit Level	RRAM Bit Level[a]	RRAM Used in Full Bit-level Mode		RRAM Used in Binary Mode	
		No Variation	With Variation	No Variation	With Variation
8 bit	7 bit	0.58%	0.58%	0.73%	0.74%
6 bit	5 bit	0.60%	0.59%		0.75%
4 bit	3 bit	0.80%	1.21%		0.75%
2 bit	1 bit	90.67%	89.10%		0.86%

- Area and Energy Estimation of Different RRAM-based Crossbar PEs



Database	Performance	CNN	BCNN	Saving
MNIST	Energy(uJ/img)	18.39	13.55	26.3%
	Area (mm^2)	0.054	0.060	-11.1%
ImageNet	Energy(uJ/img)	5444.85	2275.34	58.2%
	Area (mm^2)	21.25	9.19	56.8%

Conclusion

- In this paper, an RRAM crossbar-based accelerator is proposed for BCNN forward process.
  - The matrix splitting problem
  - The pipeline implementation.

- The robustness of BCNN on RRAM under device variation are demonstrated.
  - Experimental results show that BCNN introduces negligible recognition accuracy loss for LeNet on MNIST.
  - For AlexNet on ImageNet, the RRAM-based BCNN accelerator saves 58.2% energy consumption and 56.8% area compared with multi-bit CNN structure.

Thanks for your Attention ☺