西安電子科技大學
XIDIAN UNIVERSITY

SFU SIMON FRASER UNIVERSITY
ENGAGING THE WORLD

HyBNN: Quantifying and Optimizing Hardware Efficiency of Binary Neural Networks

Geng Yang, Jie Lei, Zhenman Fang, Yunsong li,
Jiaqing Zhang, Weiying Xie

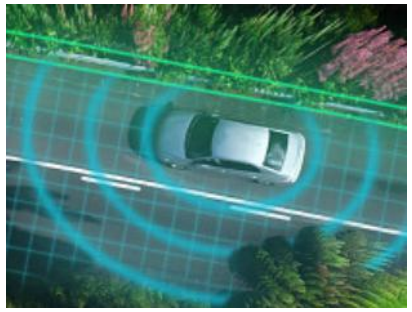
State Key Lab of Integrated Services Networks
Xidian University, China

FPT 2023

Deep Learning on the Edge……

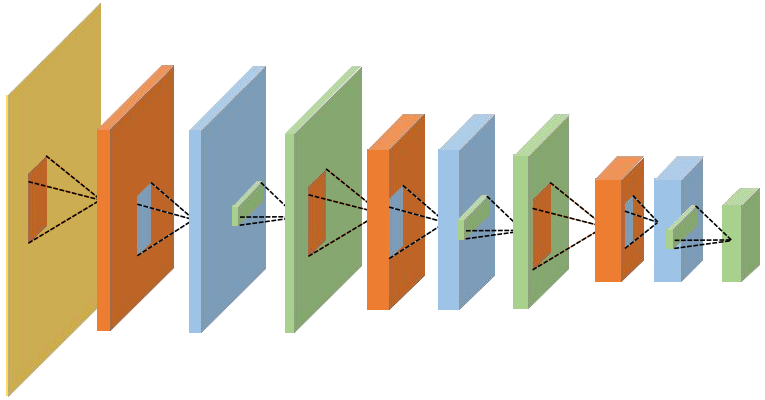
✓ Increasing demand for DNN depolyment on edge devices

- Autonomous vehicles
- Mobile phones
- Smart cities



✓ <u>Stringent deployment challenge on Edge</u>

- Deeper and more sophisticated models
- Limited memory and computing resource



<u>ResNet-18:</u>
93.5 Mb Size
1.8G MACs

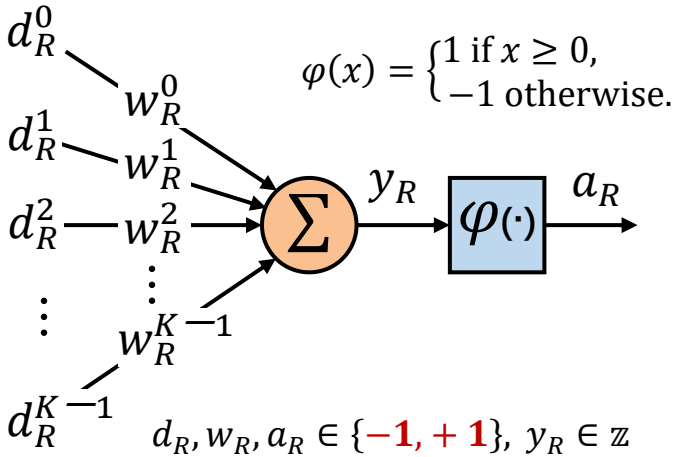
<u>AMD-Xilinx ZCU102</u>
32.1 Mb BRAM
2520 DSP

Promising Binary Neural Network

- ✓ Various model compression for edge cases
  - Network pruning
  - Knowledge disitillation
  - Compact network
  - ***Low-bit quantization***

- ✓ Promising **Binary Neural Network**
  - Extreme data precision (1W1A)
  - Smaller memory footprint
  - Cheaper XNOR-POPCNT MAC

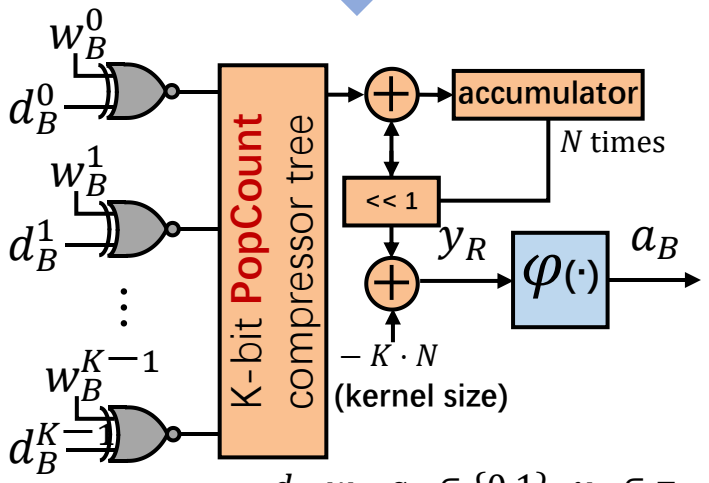
$$a_B = \varphi\left(2 \cdot PopCnt\left(\sim\left(w_B^i \hat{} d_B^i\right)\right) - K \cdot N\right)$$



Algorithm design

-1 → 0
+1 → 1

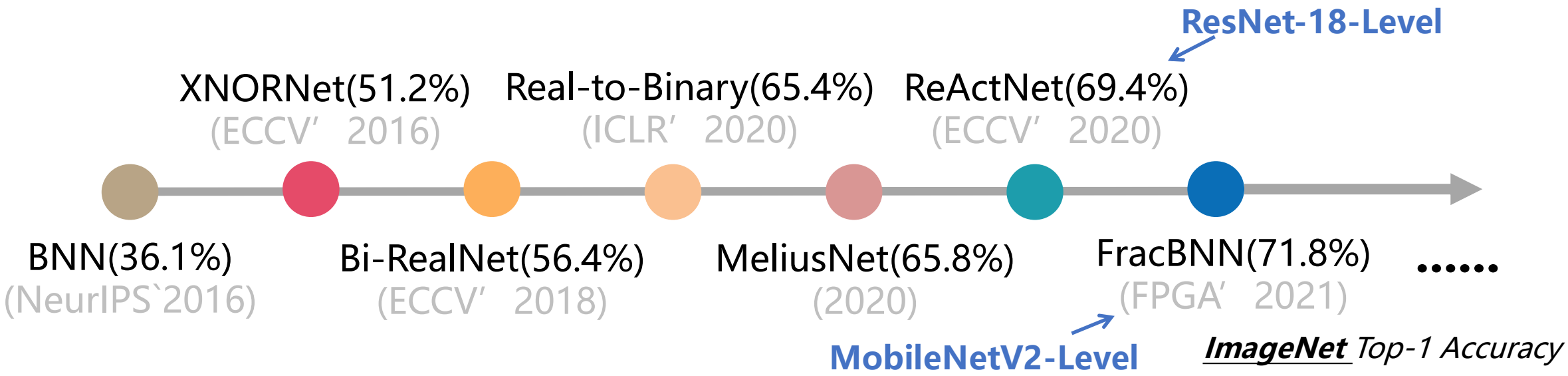
XNOR



Hardware design

Image in Barry de Bruin.,"Deep Neural Network optimization: Binary Neural Networks"

Evolution of Binary Neural Networks



However .....

✓ Satisfactory accuracy gains come at the cost of

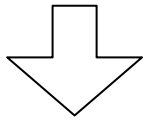
- **Various auxiliary floating-point(AFP) components**
- **Increased model size**

Main Contributions of This Paper

Our goal is to quantify such hardware inefficiency in SOTA BNNs and further optimize the BNN hardware performance with negligible accuracy loss

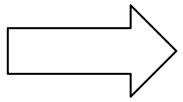
Challenge #1

Various Auxiliary Floating-point(AFP) Components



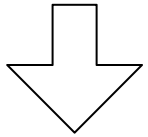
Solution #1

Algorithm/Hardware Component Fusion: **FuseBNN**



Challenge #2

Increased Model Size



Solution #2

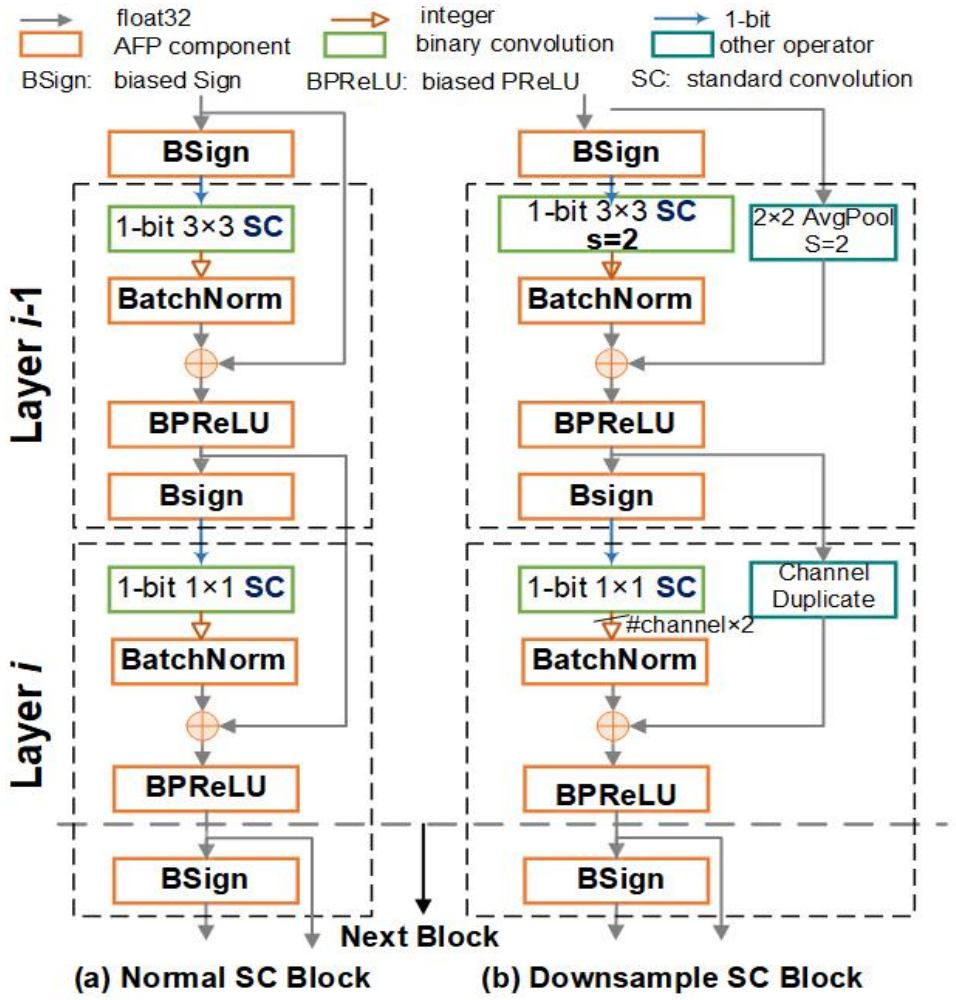
Hardware-Friendly Hybrid BNN **HyBNN**

Outline of Today’s Presentation

- **Our Case Study**
  - ✓ BaseBNN
  - ✓ BNN hardware accelerator
- **Two Challenges**
  - ✓ Various floating-point component
  - ✓ Increase model size
- **Two Solutions**
  - ✓ Algorithm/Hardware Component Fusion: FuseBNN
  - ✓ Hardware-Friendly Hybrid BNN: HyBNN
- **Experimental Results**

BNN Case Study: Ship Detection on SAR Imagery

Our goal is to quantify such hardware inefficiency in SOTA BNNs and further optimize the BNN hardware performance with negligible accuracy loss

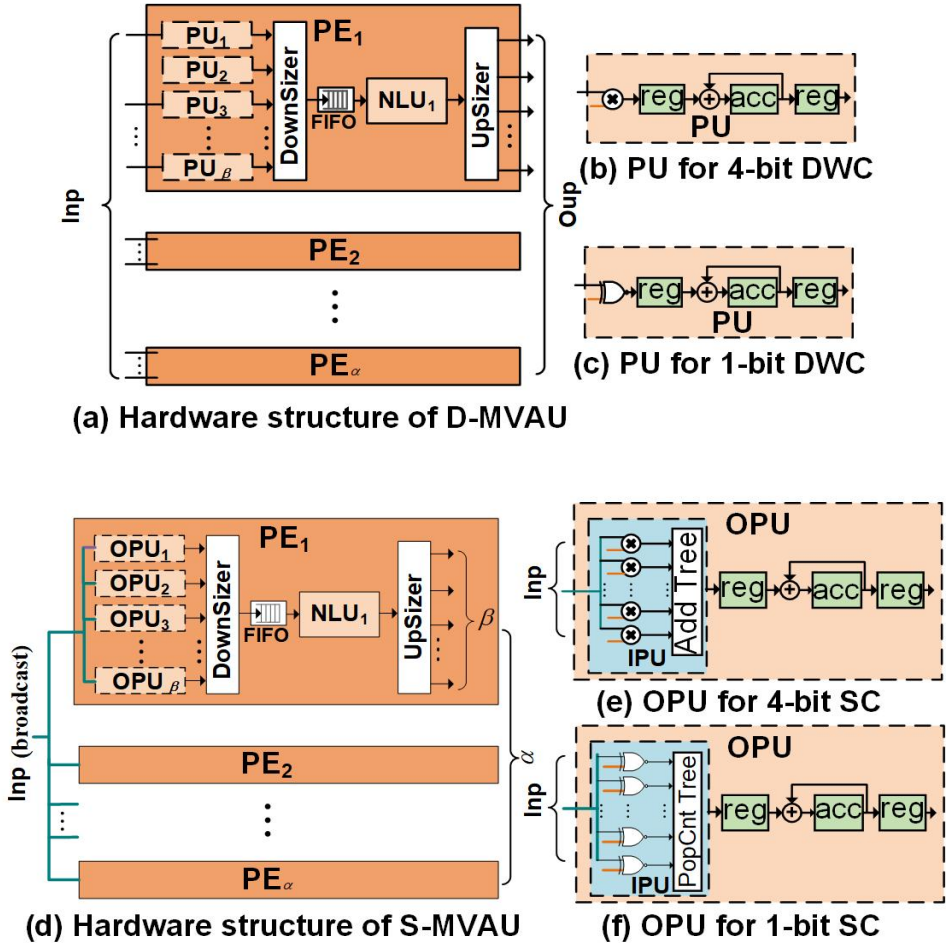


- ✓ **Our starting-point:**
  - Edge task:
    - Ship detection in SAR Imagery
  - Representive Baseline BNN
    - **ReActNet**[1]-adapted BaseBNN
    - High detection accuracy (**AP: 94.9%**)

[1] Liu et al.,"Reactnet: Towards precise binary neural network with generalized activation function," ECCV2020

BNN Case Study: Ship Detection on SAR Imagery

Our goal is to quantify such hardware inefficiency in SOTA BNNs and further optimize the BNN hardware performance with negligible accuracy loss



- ✓ **Our starting-point:**
  - Hardware Architecture
    - **All-on-chip dataflow accelerator**[2][3]
    - 4-bit accelerators for comparision
  - Hardware platform
    - AMD-Xilinx ZCU102

[2] Blott et al. "FINN-R: An end-to-end deep-learning framework for fast exploration of quantized neural networks." ACM TRETS 2018

[3] Yang et al.,"Algorithm/Hardware Codesign for Real-Time On-Satellite CNN-Based Ship Detection in SAR Imagery," IEEE TGRS2022 (open-sourced on Github)

Main Contributions of this paper

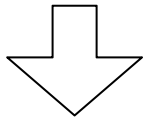
Our goal is to quantify such hardware inefficiency in SOTA BNNs and further optimize the BNN hardware performance with negligible accuracy loss

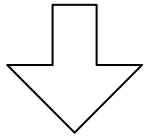
Challenge #1

Various Auxiliary Floating-point(AFP) Components

Challenge #2

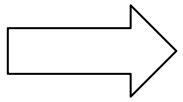
Increased Model Size





Solution #1

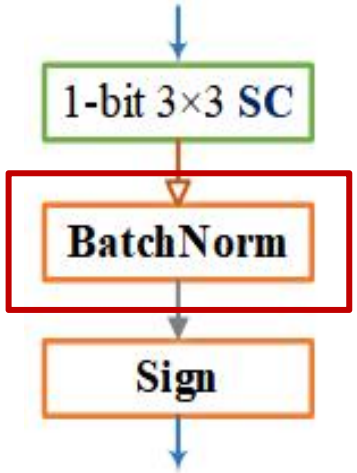
Algorithm/Hardware Component Fusion: **FuseBNN**



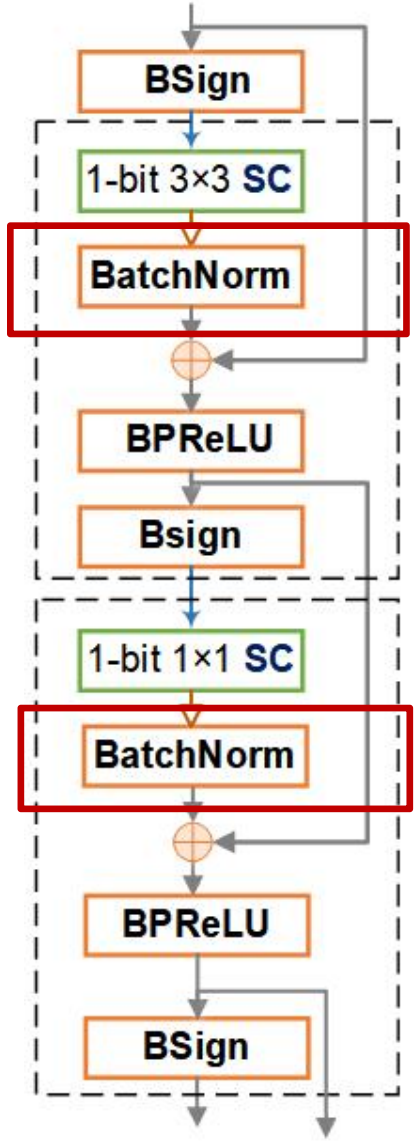
Solution #2

Hardware-Friendly Hybrid BNN **HyBNN**

Challenge #1: Auxiliary Floating-point Components



Early-stage BNN
low accuracy



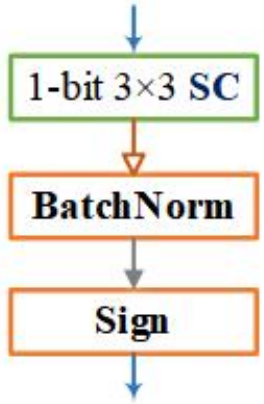
SOTA BNN: BaseBNN
high accuracy

✓ **BatchNorm**

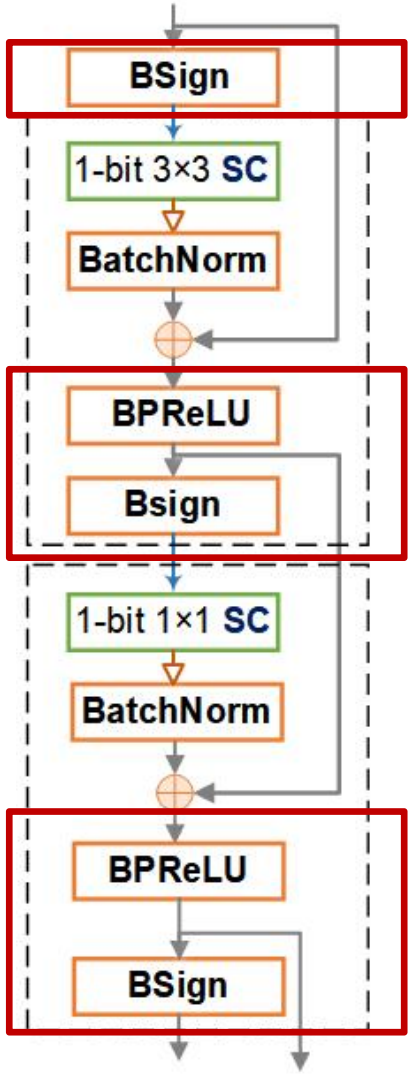
$$y_j = k_j x_j + b_j$$

$$k_j = \frac{\gamma_j}{\sqrt{||\sigma^2{}_j + \varepsilon||}}, \quad b_j = \beta_j - \frac{\gamma_j \mu_j}{\sqrt{||\sigma^2{}_j + \varepsilon||}}$$

Challenge #1: Auxiliary Floating-point Components



Early-stage BNN
low accuracy



SOTA BNN: BaseBNN
high accuracy

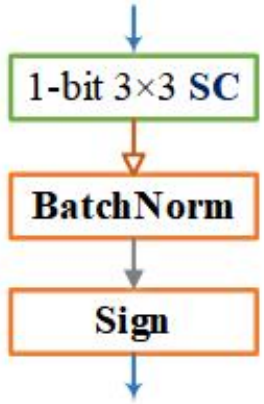
- ✓ BatchNorm
- ✓ **Biased PReLU/Biased Sign**[1]

$$BPReLU(m_j) = \begin{cases} m_j + \phi_j + \xi_j & \text{if } m_j > -\phi_j \\ \lambda_j(m_j + \phi_j) + \xi_j & \text{if } m_j \le -\phi_j \end{cases}$$

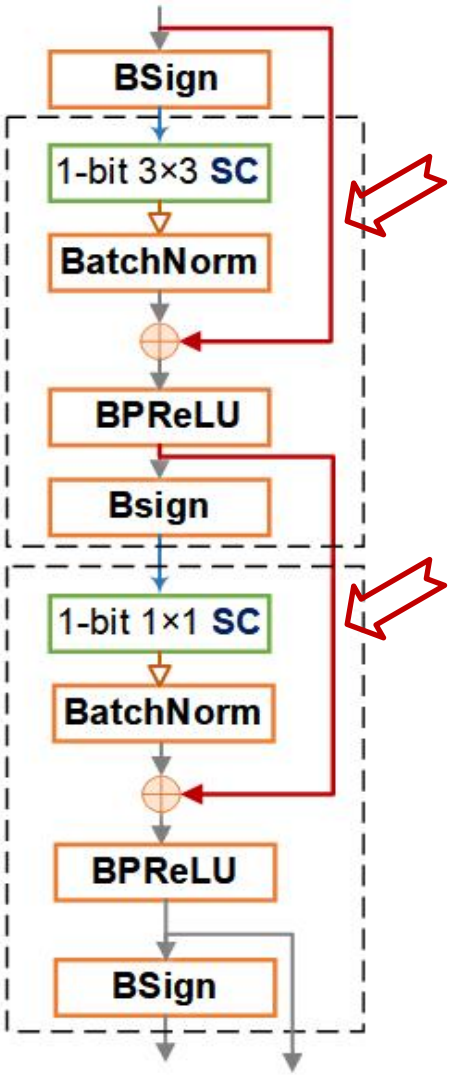
$$x_j^b = Bsign(x_j^r) = \begin{cases} +1 & \text{if } x_j^r > -\omega_j \\ -1 & \text{if } x_j^r \le -\omega_j \end{cases}$$

[1] Liu et al.,"Reactnet: Towards precise binary neural network with generalized activation function," ECCV2020

Challenge #1: Auxiliary Floating-point Components



Early-stage BNN
low accuracy



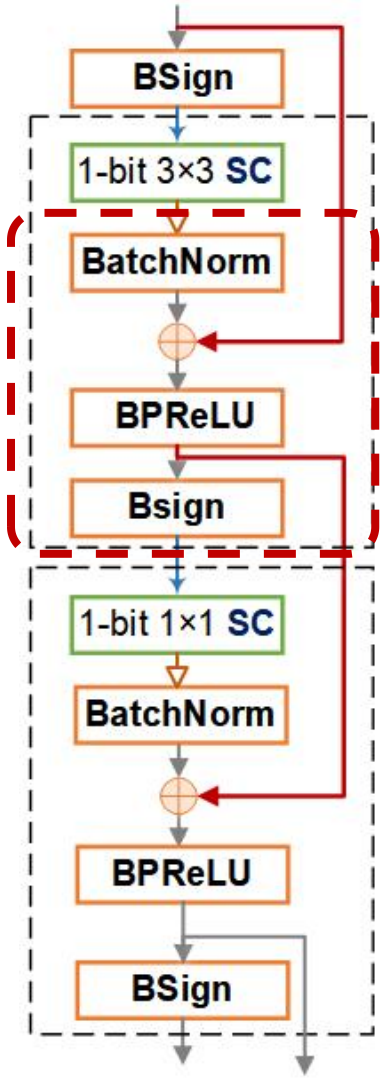
SOTA BNN: BaseBNN
high accuracy

- ✓ BatchNorm
- ✓ Biased PReLU/Biased Sign[1]
- ✓ **Shortcut branch**

(widely used for accuracy improvement)

[1] Liu et al.,"Reactnet: Towards precise binary neural network with generalized activation function," ECCV2020

Challenge #1: Auxiliary Floating-point Components

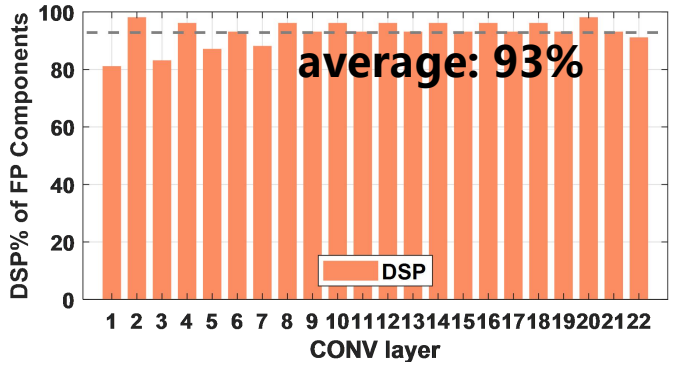


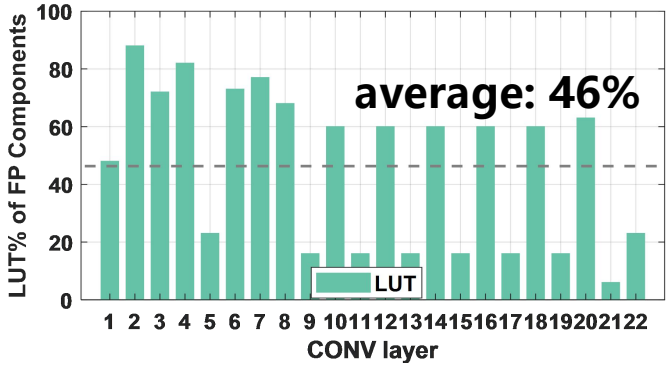
Threshold Unit

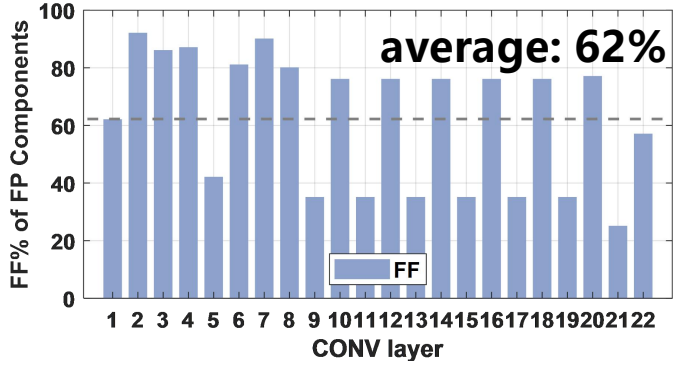
mapping

SOTA BNN: BaseBNN

high accuracy







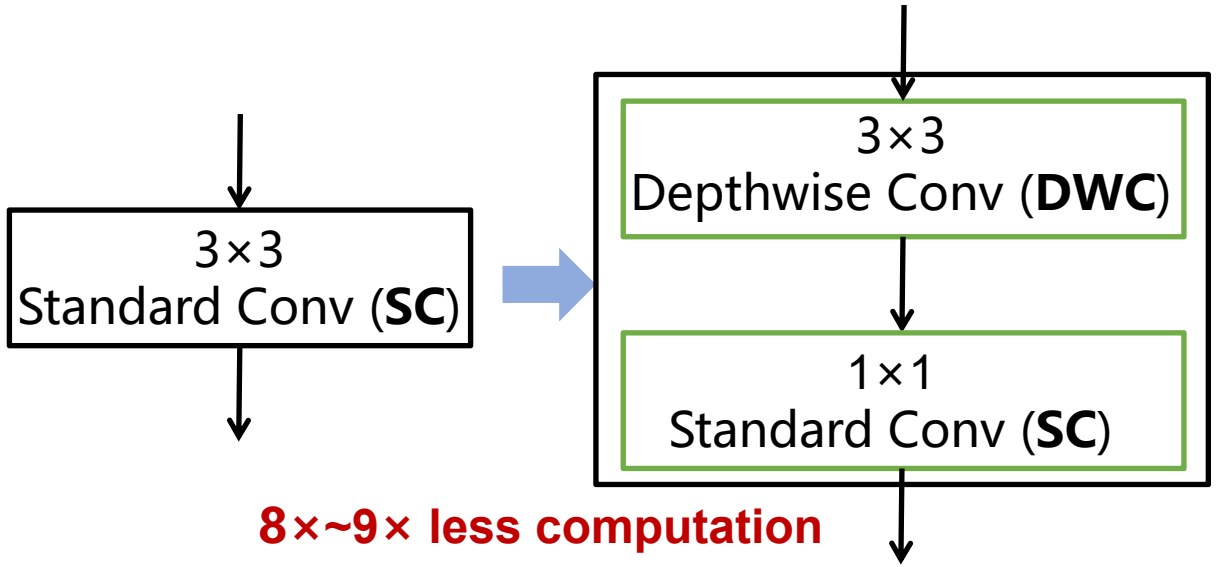
Percentage of AFP component resource consumption in each convolution layer of BaeseBNN

- ✓ Single threshold unit consumes **5 DSP, 820 FFs and 517 LUTs**
- ✓ An average of **93% DSPs, 46% LUTs and 62% FFs**

Challenge #2: Increased Model Size

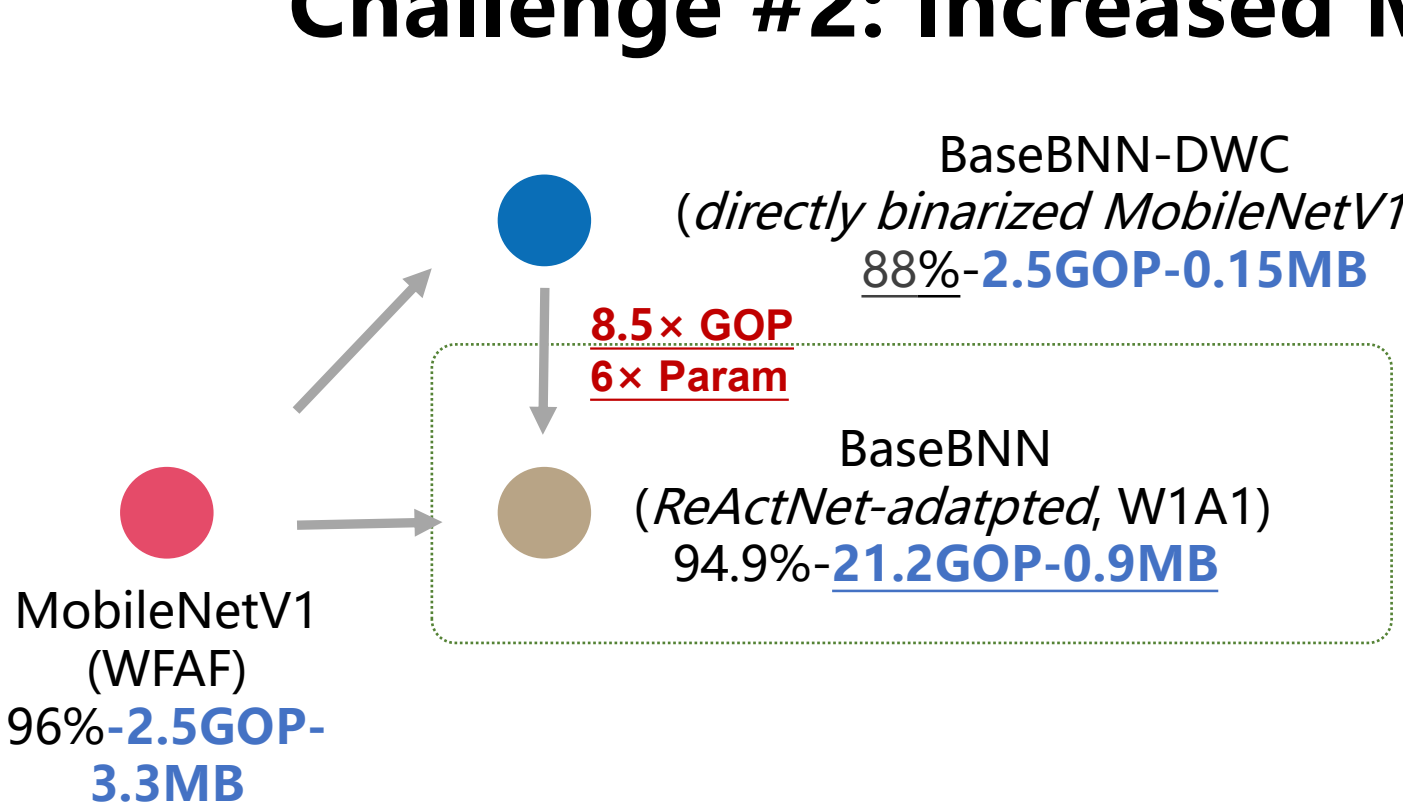
BaseBNN-DWC
(*directly binarized MobileNetV1*,W1A1)
88%-**2.5GOP-0.15MB**

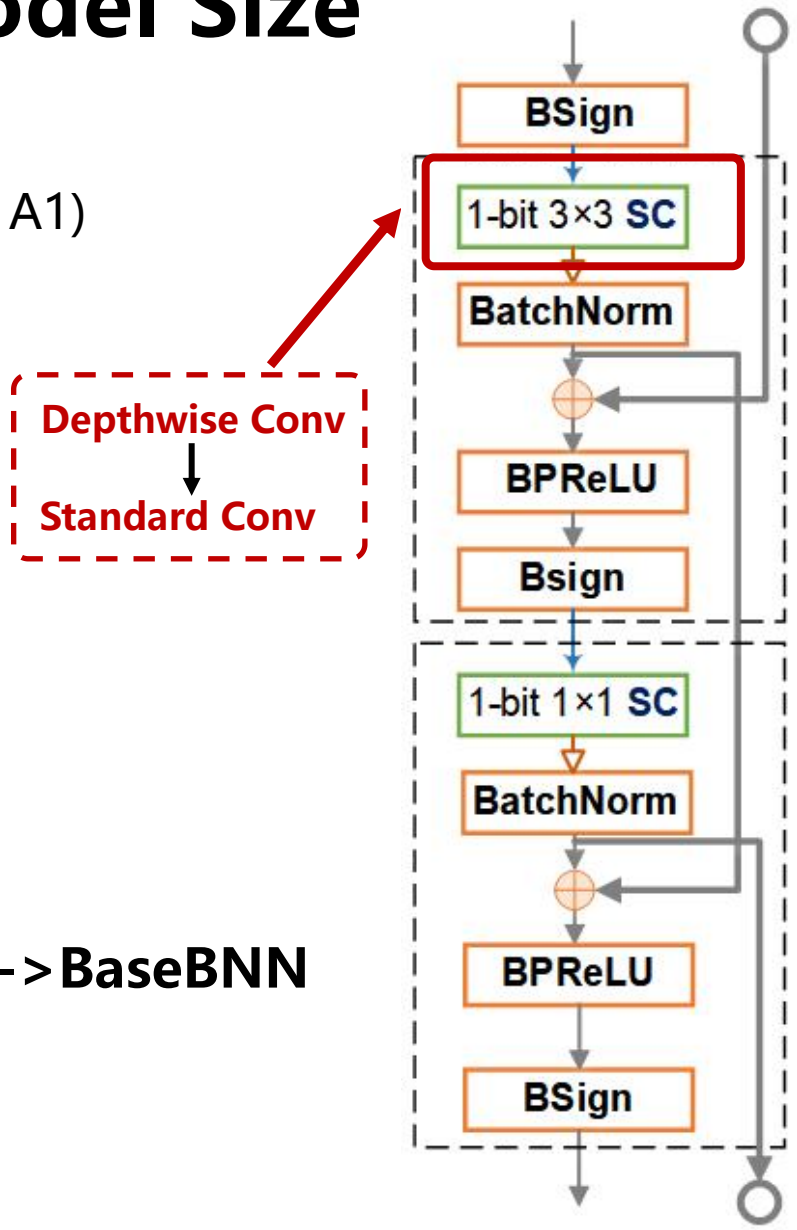
MobileNetV1[4]
(WFAF)
96%-**2.5GOP-3.3MB**



[4] Howard et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." ar**v preprint ar**v:1704.04861, 2017.

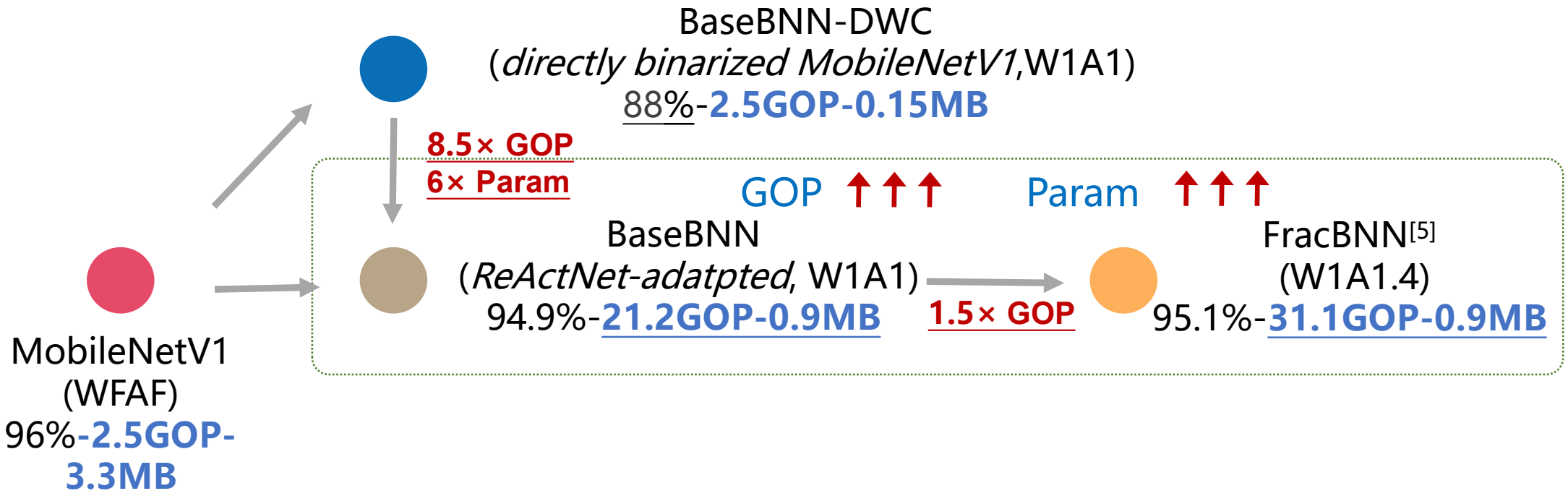
Challenge #2: Increased Model Size





✓ More binary standard convolution **BaseBNN-DWC->BaseBNN**

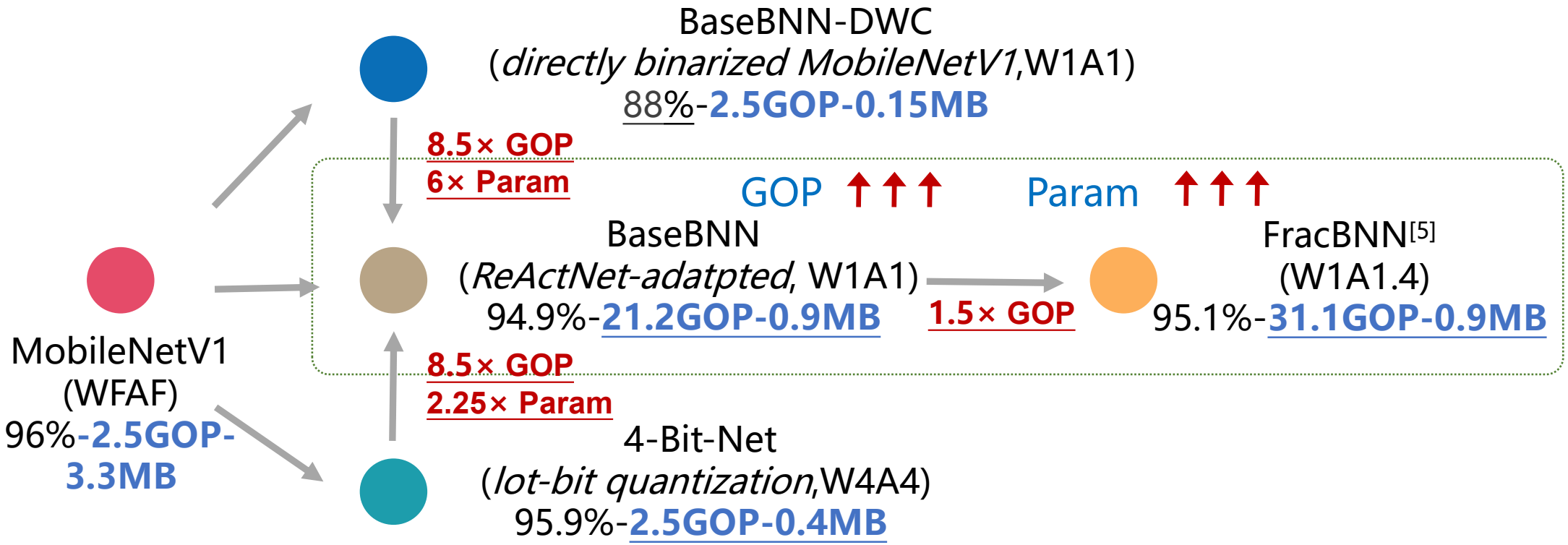
Challenge #2: Increased Model Size



- ✓ More binary standard convolution **BaseBNN-DWC->BaseBNN**
- ✓ More complex fractional convolution **BaseBNN->FracBNN[2]**

[5] Zhang et al.,"Fracbnn: Accurate and fpga-efficient binary neural networks with fractional activations," FPGA2021

Challenge #2: Increased Model Size



- ✓ More binary standard convolution **BaseBNN-DWC->BaseBNN**
- ✓ More complex fractional convolution **BaseBNN->FracBNN[2]**
- ✓ losing advantage over 4-Bit-Net **BaseBNN->4-Bit-Net**

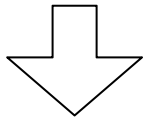
[5] Zhang et al.,"Fracbnn: Accurate and fpga-efficient binary neural networks with fractional activations," FPGA2021

Main Contributions of this paper

Our goal is to quantify such hardware inefficiency in SOTA BNNs and further optimize the BNN hardware performance with negligible accuracy loss

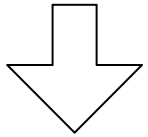
Challenge #1

Various Auxiliary Floating-point(AFP) Components



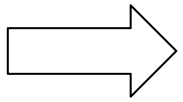
Challenge #2

Increased Model Size



Solution #1

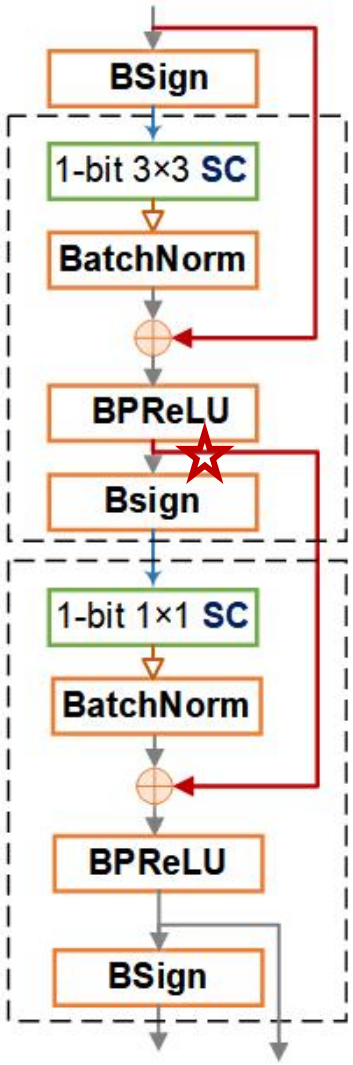
Algorithm/Hardware Component Fusion: **FuseBNN**



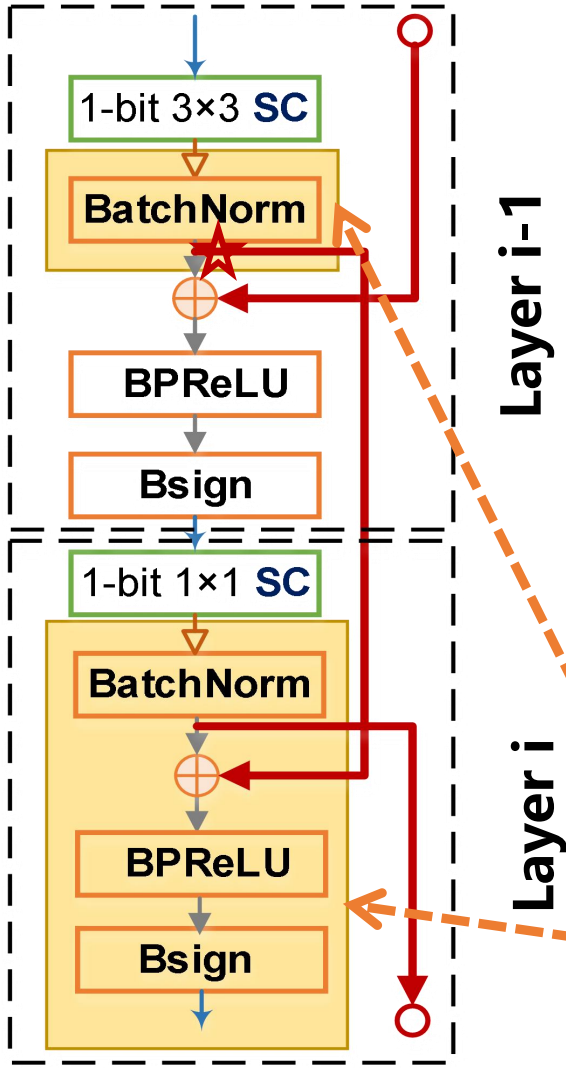
Solution #2

Hardware-Friendly Hybrid BNN **HyBNN**

FuseBNN: Algorithm/Hardware Component Fusion



SOTA BNN:
BaseBNN



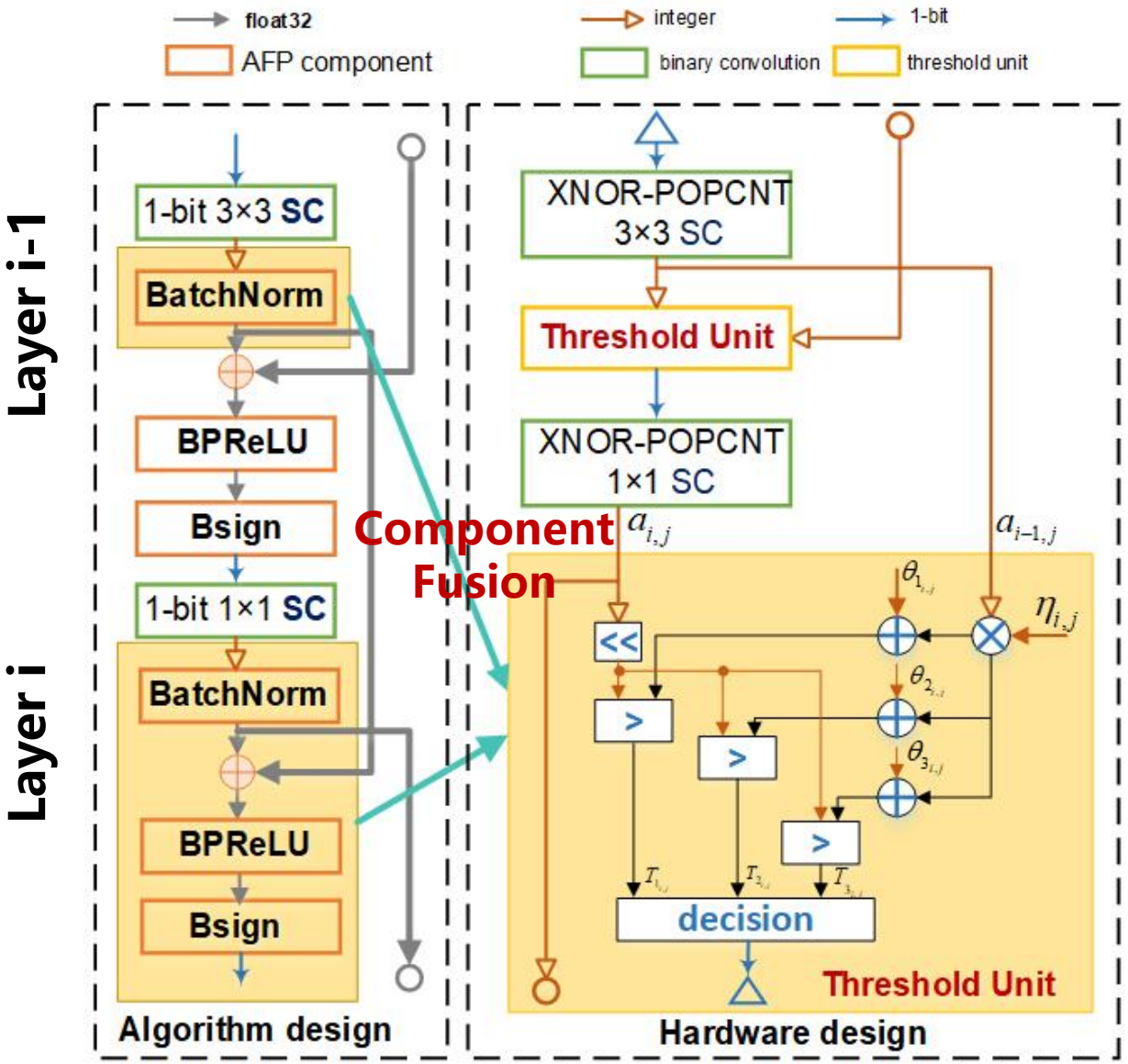
Our FuseBNN

Solution to Challenge #1

- ✓ Fuse (and retain) all axuliary floating-point components
- ✓ **Light change in AFP shortcut branch**
- ✓ Without accuracy loss

Component Fusion

FuseBNN: Algorithm/Hardware Component Fusion



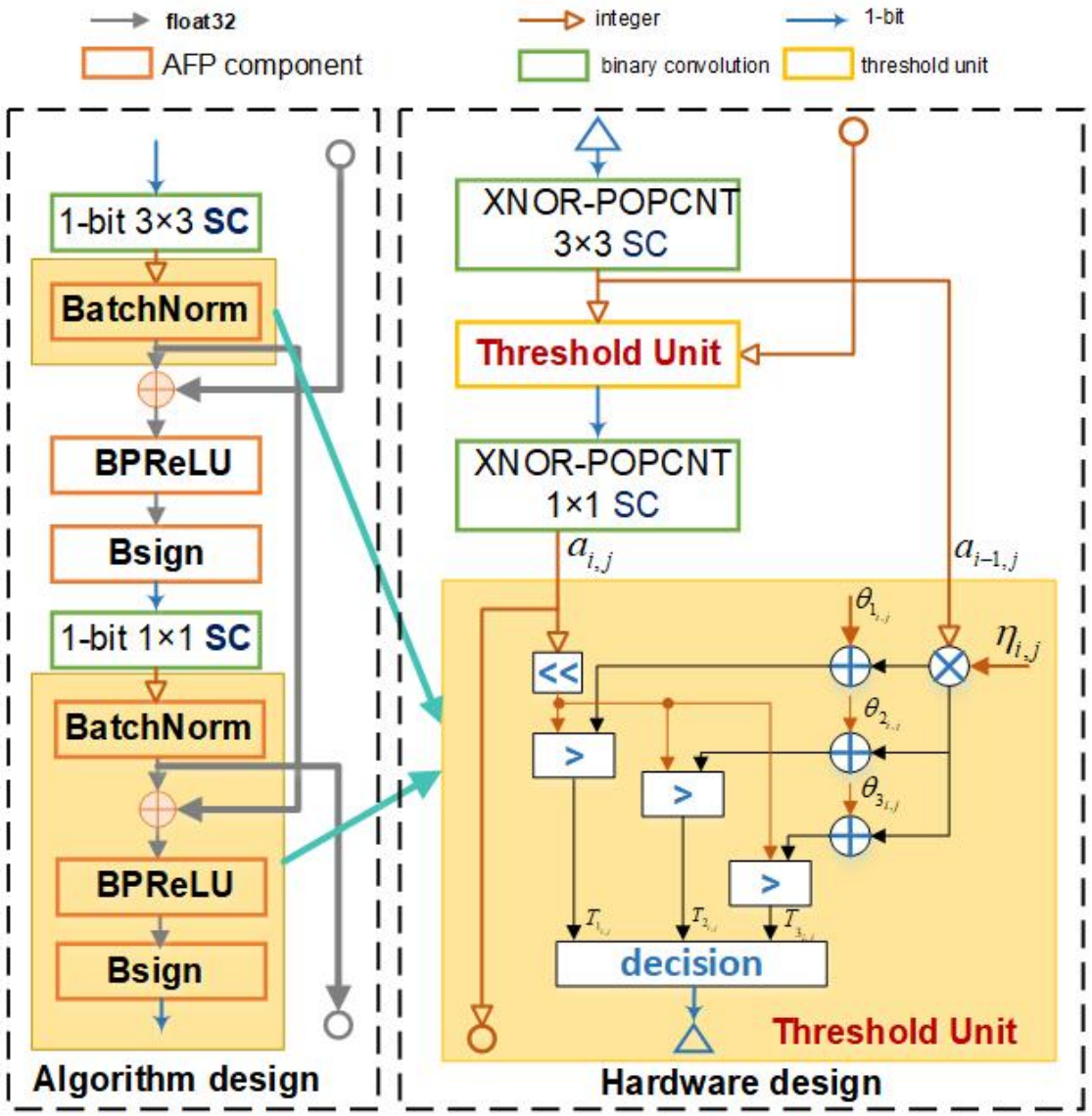
Solution to Challenge #1

- ✓ Single fused threshold unit
  - Pre-computed constant coefficients
  - 1 multiplication, 3 additions and several logical decision

$$o_{i,j} = \begin{cases} 1 & \text{if } ((\text{cond1}\&\text{cond2}) \mid (\text{cond3}\&\text{cond4})) \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} \text{cond1}: &\quad a_{i,j} > \eta_{i,j} a_{i-1,j} + \theta_{1_{i,j}} \\ \text{cond2}: &\quad a_{i,j} > \eta_{i,j} a_{i-1,j} + \theta_{2_{i,j}} \\ \text{cond3}: &\quad a_{i,j} \le \eta_{i,j} a_{i-1,j} + \theta_{1_{i,j}} \\ \text{cond4}: &\quad a_{i,j} > \eta_{i,j} a_{i-1,j} + \theta_{3_{i,j}} \end{aligned}$$

FuseBNN: Algorithm/Hardware Component Fusion

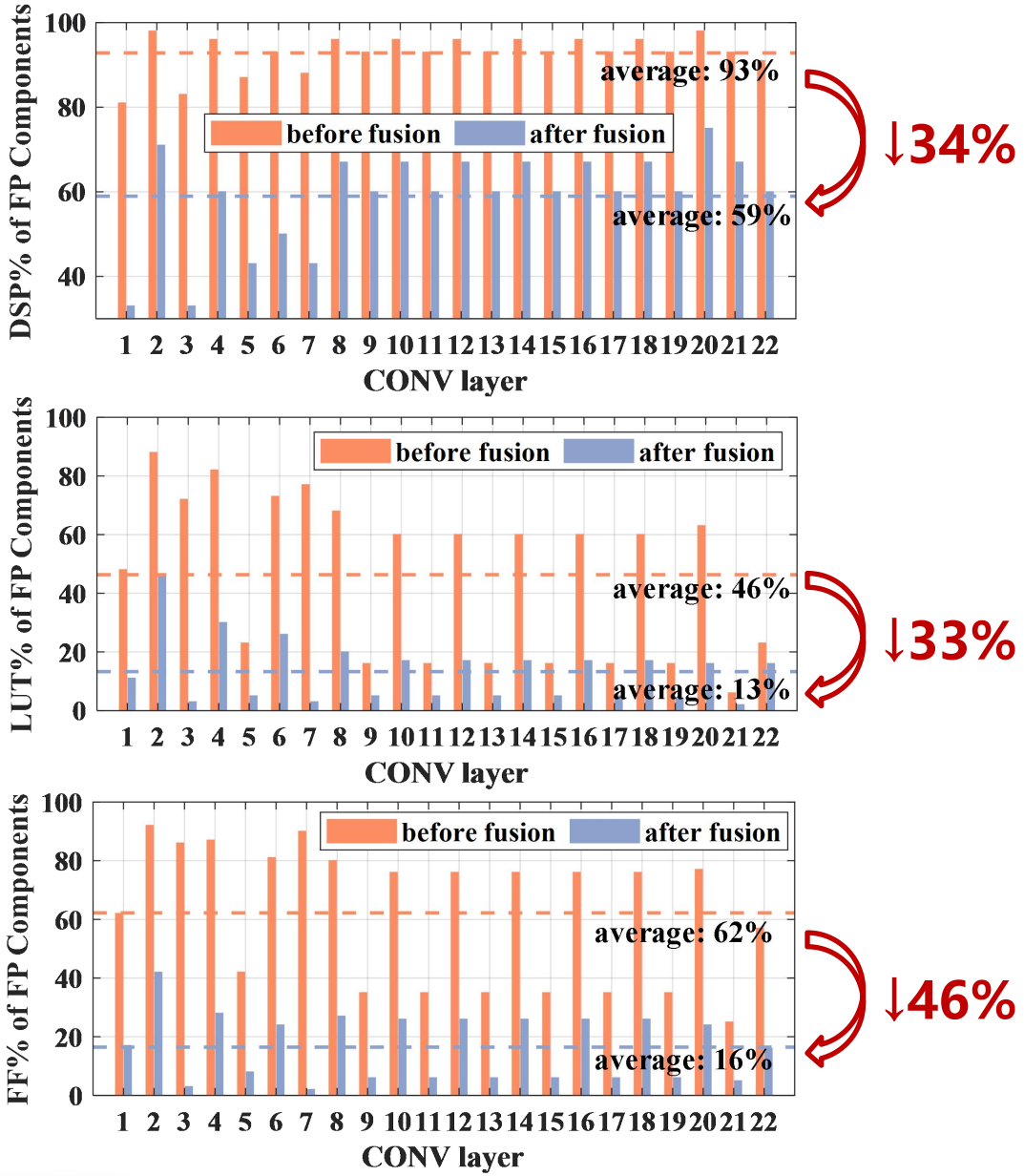


Solution to Challenge #1

- ✓ Single fused threshold unit
  - Pre-computed constant coefficients
  - 1 multiplication, 3 additions and several logical decision
  - **5×DSP，20.5×FF, 6.5×LUT reduction**

Threshold Unit	DSP	FF	LUT
Before fusion	5	820	517
After fusion	**1**	**40**	**80**

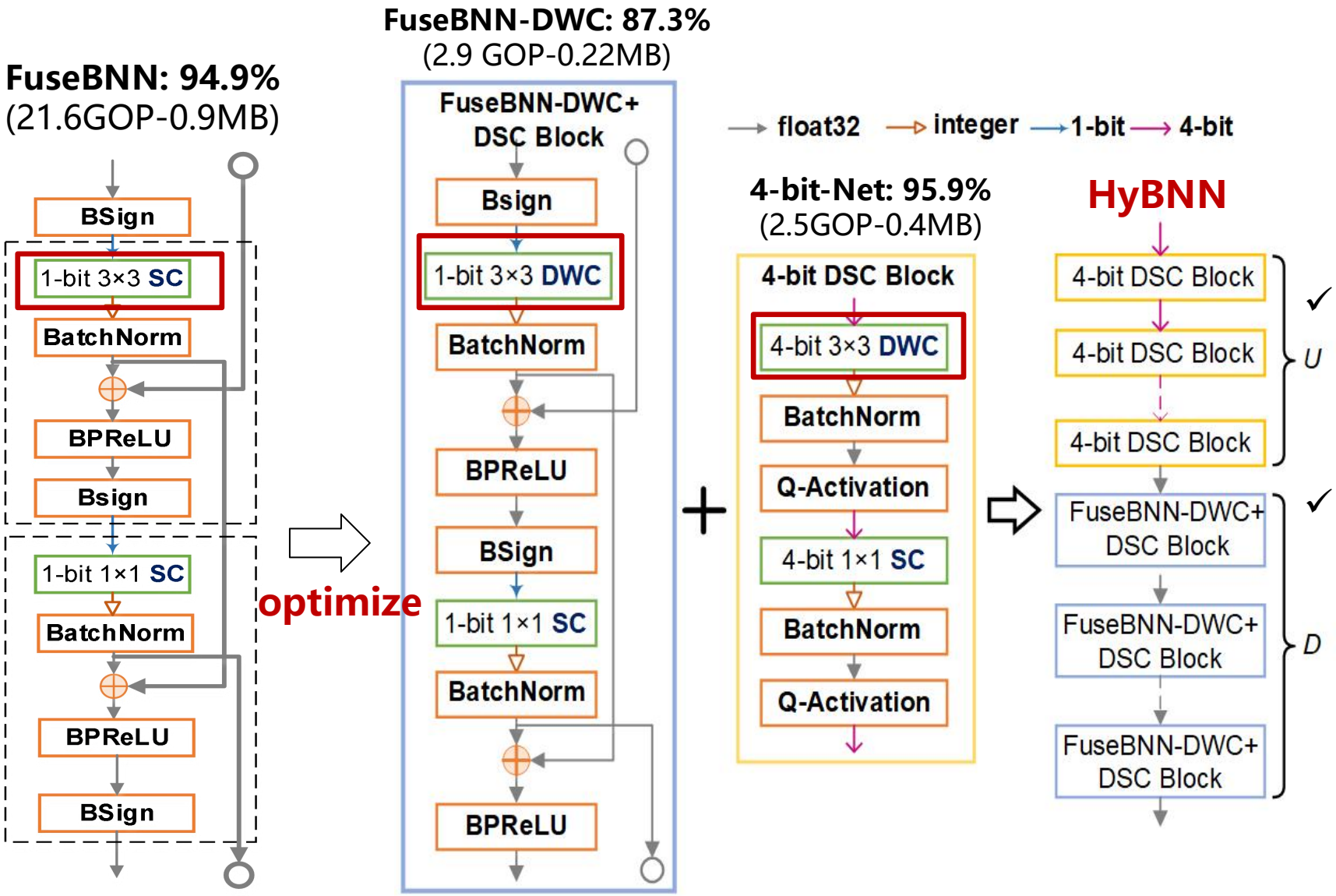
FuseBNN: Algorithm/Hardware Component Fusion



Solution to Challenge #1

- ✓ Single fused threshold unit
  - Pre-computed constant coefficients
  - 1 multiplication, 3 additions and several logical decision
- ✓ An average reduction of **34% DSPs, 33% LUTs and 46% FFs**

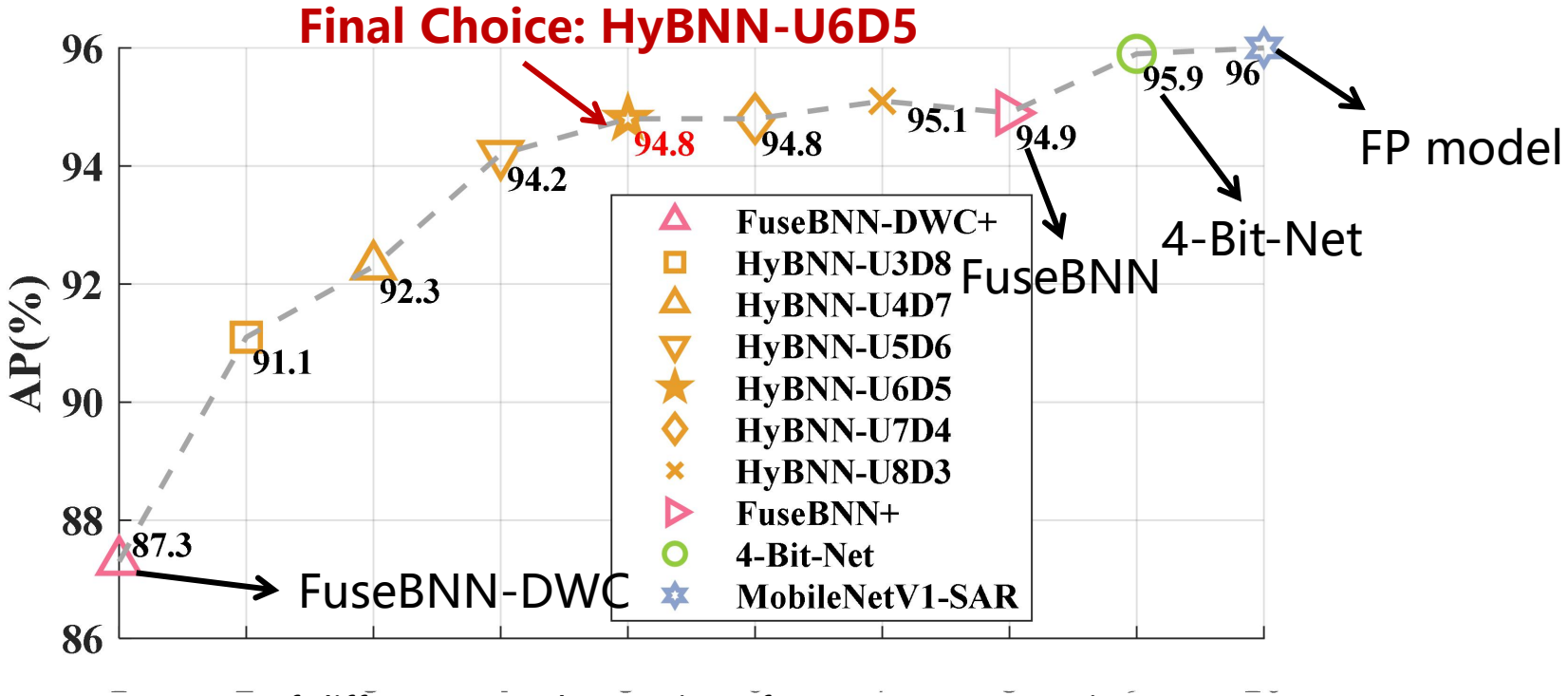
HyBNN: Hardware-Friendly Hybrid BNN



Solution to Challenge #2

- ✓ Directly binarize the DSC blocks to keep its compact model size **for the first time**.
- ✓ Use 4-Bit-Net to compensate for the accuracy loss

Experimental Result of HyBNN Accuracy



Accuracy of different U and D settings for HyBNN on SAR imagery

✓ HyBNN with U-D setting

- Trade off between accuacy and resource overhead
- **HyBNN-U6D5 (94.8 AP)**

Experimental Result for SAR ship detection

Model	AP (%)	GOP	Param (MB)	Resource Cost				Peak FPS	FPS /BRAM	FPS /DSP	FPS /kLUTs
				BRAM	DSP	FF	LUT				
BaseBNN	94.9	21.2	0.9	**1,032.5 (113.2%)**	1,146 (45.5%)	472,534 (86.2%)	**399,544 (145.8%)**	**failed**	-	-	-
FuseBNN+	94.9	21.6	0.9	**810 (88.8%)**	162 (6.4%)	168,688 (30.8%)	122,129 (44.6%)	**90**	0.11	0.56	0.74
4-Bit-Net (250MHz)	95.9	2.5	0.4	466 (51.1%)	1,997 (79.2%)	367,112 (67.0%)	192,157 (70.1%)	**230**	0.49	0.12	1.20

Performance Report on AMD-Xilinx ZCU102 FPGA with 300MHz (Input size: 416×416)

- ✓ FuseBNN+: lower DSP, FF, LUT usage compared to BaseBNN
- ✓ **However, increased model size still make FuseBNN lose advantage over 4-Bit-Net**

Experimental Result for SAR ship detection

Model	AP (%)	GOP	Param (MB)	Resource Cost				Peak FPS	FPS /BRAM	FPS /DSP	FPS /kLUTs
				BRAM	DSP	FF	LUT				
BaseBNN	94.9	21.2	0.9	1,032.5 (113.2%)	1,146 (45.5%)	472,534 (86.2%)	399,544 (145.8%)	failed	-	-	-
FuseBNN+	94.9	21.6	0.9	810 (88.8%)	162 (6.4%)	168,688 (30.8%)	122,129 (44.6%)	90	0.11	0.56	0.74
4-Bit-Net (250MHz)	95.9	2.5	0.4	466 (51.1%)	1,997 (79.2%)	367,112 (67.0%)	192,157 (70.1%)	230	0.49	0.12	1.20
HyBNN	94.8	**2.5**	**0.19**	555 (60.9%)	1,662 (66.0%)	276,583 (50.5%)	152,687 (55.7%)	**615**	**1.11**	0.37	**4.03**

Performance Report on AMD-Xilinx ZCU102 FPGA with 300MHz (Input size: 416×416)

- ✓ HyBNN: **higher FPS/BRAM(2.3×), FPS/DSP(3.1×), FPS/kLUTs(3.4×)** efficiency over 4-Bit-Net

Generalization Study for CIFAR10

Model	Top-1 (%)	GOP	Param (MB)	Resource Cost				Peak FPS	FPS /BRAM	FPS /DSP	FPS /kLUTs
				BRAM	DSP	FF	LUT				
HyBNN (300MHz)	**89.8**	0.03	0.07	94.5 (43.8%)	205 (56.9%)	99,074 (70.2%)	51,927 (73.6%)	**4302**	**45.5**	21	**82.8**
FracBNN[5] (250MHz)	89.1	0.07	0.03	212 (98.1%)	126 (35%)	39,618 (28.1%)	51,444 (72.9%)	2806	13.2	22.3	54.5

Performance Comparision to SOTA FracBNN on AMD-Xilinx Ultra96-V2

- ✓ GOP reduction (**2.3×**)
- ✓ **HyBNN** achieve better accuracy (**0.7%**), higher higher FPS (**1.5×**), higher FPS/BRAM(**3.4×**), FPS/kLUTs(**1.5×**) efficiency over SOTA FracBNN

[5] Zhang et al.,"Fracbnn: Accurate and fpga-efficient binary neural networks with fractional activations," FPGA2021

Key Take-Aways

- ✓ **A quantitative evaluation** of hardware inefficiency caused by AFP components and increased model size in SOTA BNNs
- ✓ **FuseBNN**, a novel algorithm/hardware co-design to fuse AFP operators in BNNs
- ✓ **HyBNN**, the first hybrid BNN and 4-Bit-Net design that directly binarizes (and quantizes) the original DSC blocks
- ✓ Promising experimental results for **ship detection on SAR imagery and image classification on CIFAR-10** on embeded FPGA

- ✓ Future work: We plan to explore more SOTA BNNs, datasets, and FPGAs.

西安電子科技大學
XIDIAN UNIVERSITY

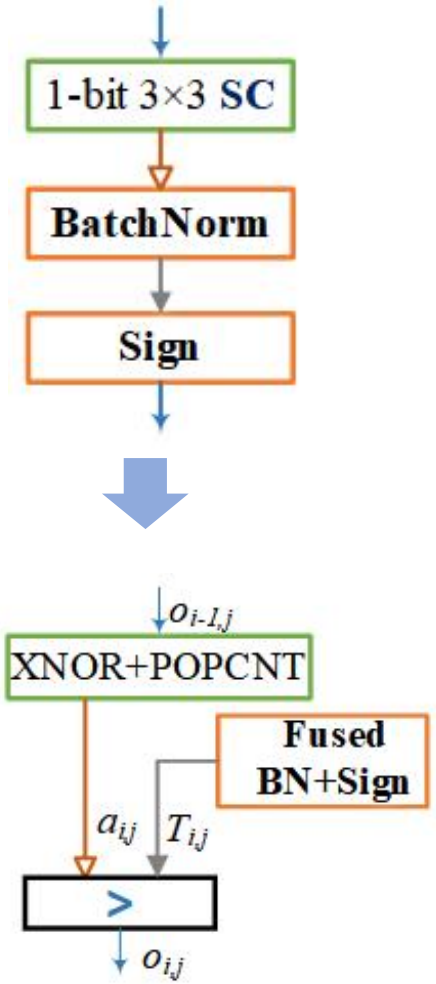
Thank You! Questions?

Email: gengyang@stu.xidian.edu.cn

HyBNN: Quantifying and Optimizing Hardware Efficiency of Binary Neural Networks,
Geng Yang, Jie Lei, Zhenman Fang, Yunsong li, Jiaqing Zhang, Weiying Xie

FuseBNN: Algorithm/Hardware Component Fusion

early-stage BNN
low accuracy

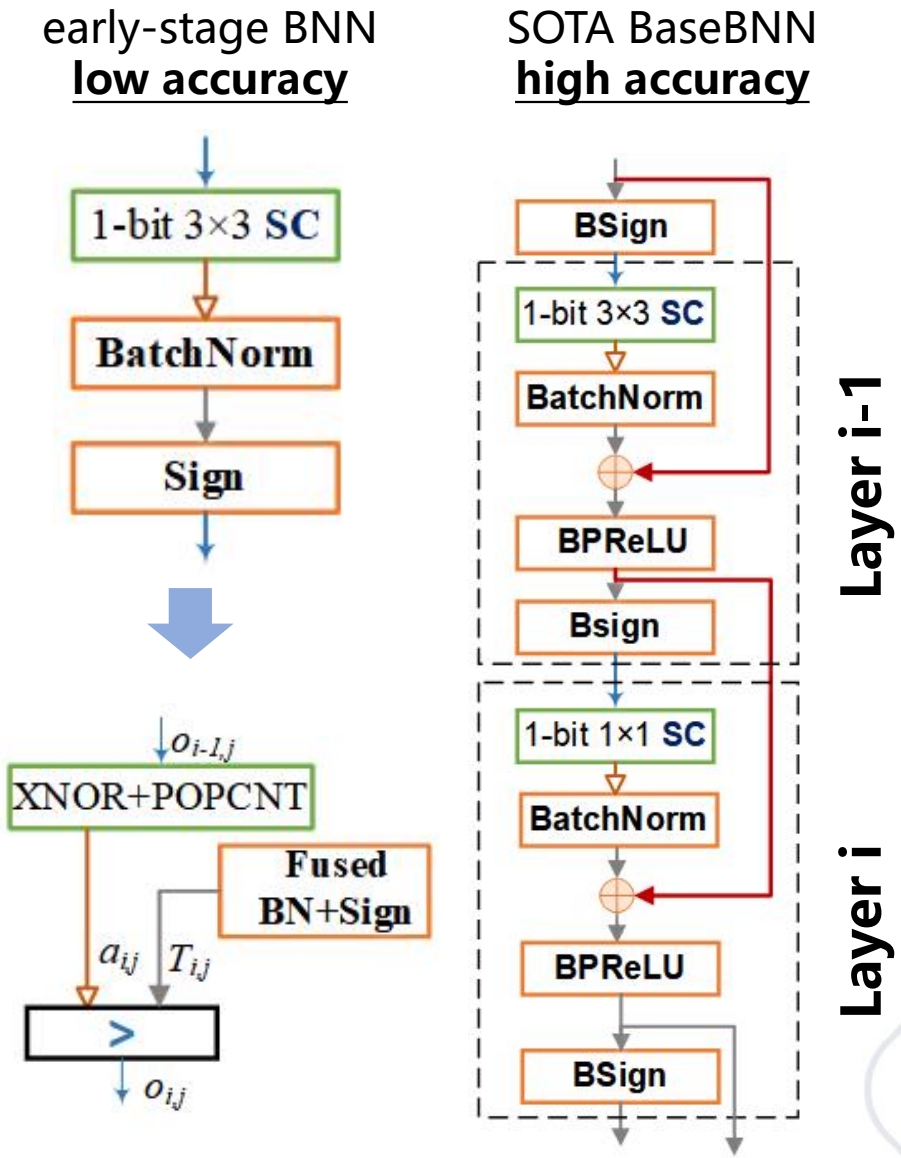


Solution to Challenge #1

- ✓ Early-stage BNN
  - cheap fusion threshold unit

$$o_{i,j} = \begin{cases} 1 & if\ a_{i,j} > T_{i,j} \\ 0 & otherwise \end{cases}, \quad T_{i,j} = -\frac{b_{i,j}}{2k_{i,j}} + \frac{N_i}{2}$$

FuseBNN: Algorithm/Hardware Component Fusion



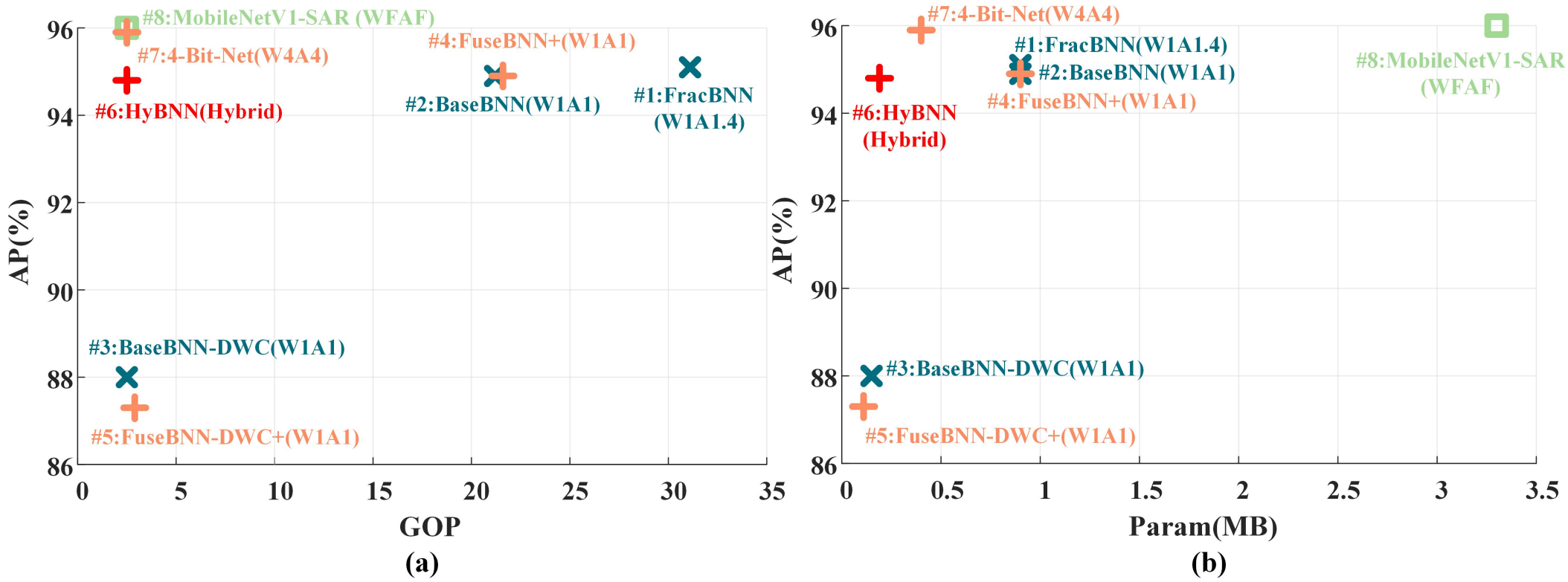
Solution to Challenge #1

- ✓ Early-stage BNN
  - cheap fusion threshold unit

$$o_{i,j} = \begin{cases} 1 & if\ a_{i,j} > T_{i,j} \\ 0 & otherwise \end{cases}, \quad T_{i,j} = -\frac{b_{i,j}}{2k_{i,j}} + \frac{N_i}{2}$$

- ✓ **the introduction of floating-point shortcut branch break simple fusion strategy**

Model Size Analysis



Accuracy (AP), Operation (GOP, giga multiply-accumulate operations, not considering bit-width) and parameter size (MB) comparsion for different models. WFAF denotes 32-bit floating-point weight and activation

Model Configuration

Table 1. Configuration of backbone network MobileNetV1-SAR. Each row describes a sequence of n (last column) repeated identical layers. The first column shows the input feature map size for each operator (second column). c and s denote the number of output channels and stride of each operator.

Input	Operator		c	s	n
416×416×1	3×3 SC		32	2	1
208×208×32	DSC	3×3 DWC	32	1	1
		1×1 SC	64	1	
208×208×64	DSC	3×3 DWC	64	2	1
		1×1 SC	128	1	
104×104×128	DSC	3×3 DWC	128	1	1
		1×1 SC	128	1	
104×104×128	DSC	3×3 DWC	128	2	1
		1×1 SC	256	1	
52×52×256	DSC	3×3 DWC	256	1	5
		1×1 SC	256	1	
52×52×256	DSC	3×3 DWC	256	1	1
		1×1 SC	512	1	
52×52×512	DSC	3×3 DWC	512	1	1
		1×1 SC	512	1	
52×52×512	1×1 SC		25	1	1
52×52×25	detector		-	-	-

Ship detection for SAR Imagery

Table 5. Configuration of backbone network MobileNetV1-CIFAR-10.

Input	Operator	c	s	n
32×32×3	SC 3×3	32	1	-
32×32×32	DWC 3×3	32	1	1
	SC 1×1	64	1	
32×32×64	DWC 3×3	64	2	1
	SC 1×1	128	1	
16×16×128	DWC 3×3	128	1	2
	SC 1×1	128	1	
16×16×128	DWC 3×3	128	2	1
	SC 1×1	256	1	
8×8×256	DWC 3×3	256	1	3
	SC 1×1	256	1	
8×8×256	Max_Pool	256	-	3
1×1×256	FC	10	-	-
1×1×10	Softmax	-	-	-

Image Classification for CIFAR10

Experimental Result for CIFAR10

Table 6. Operation number (GOP, *not considering bit-width*), parameter size (MB) and accuracy comparison for image classification on CIFAR-10.

Model	Param (MB)	GOP	Top-1 (%)
BaseBNN	0.32	0.25	89.8
FuseBNN			90
FuseBNN-DWC+	0.04	**0.03**	77.5
HyBNN-U5D3	**0.07**		**89.8**
4-Bit-Net	0.14		90.1
FracBNN[30]	0.03	0.07	89.1