INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING
ICPP/2021/CHICAGO/USA
acm In-Cooperation
sighpc
AUGUST 9-12, 2021

BitX: Empower Versatile Inference with Hardware Runtime Pruning

Hongyan Li[12], Hang Lu[12], Jiawen Huang[1], Wenxu Wang[12], Mingzhe Zhang[1], Wei Chen[1], Liang Chang[3], Xiaowei Li[12]

[1]State Key Laboratory of Computer Architecture, Institute of Computing Technology, CAS, Beijing, China

[2]University of Chinese Academy of Sciences, Beijing, China

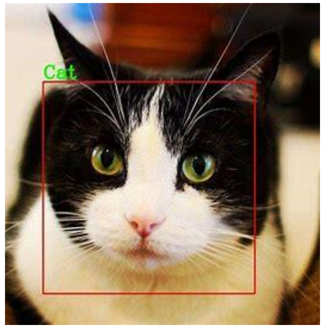
[3]University of Electronic Science and Technology of China, Chengdu, China

计算机体系结构国家重点实验室
State Key Laboratory of Computer Architecture, ICT, CAS

中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

Introduction

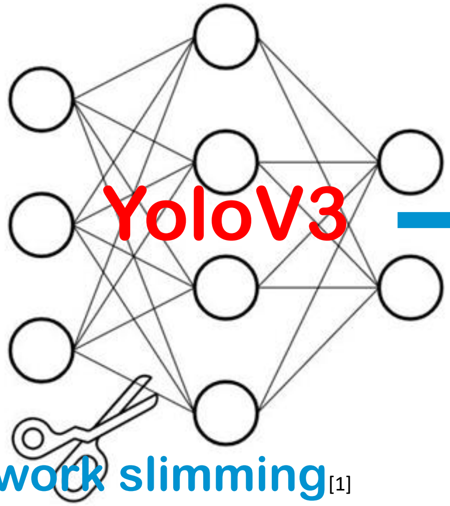
DNN networks



Model	Params
ResNet50	22.5×10^6
3D-ConvNet	79×10^6
Bert_large	340×10^6

Demand of large computation

Pruning



network slimming[1]

2~3 days sparse training

Time-consuming

3~4 days retraining / fine-tuning

[1] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui,Zhang. 2017. Learning Efficient Convolutional Networks through Network Slimming. In ICCV

Introduction

The contribution of this work

Propose a novel hardware runtime pruning method -- BitX, to empower versatile DNN inference

① Software effortless — No retraining! No fine-tuning!

② Orthogonal to the existing software pruning methodologies — Obtain additional speedup

③ Multi-precision support — Floating point & fixed point DNNs

Introduction

The contribution of this work

Propose a deep learning accelerator capable of unprecedented hardware runtime pruning to mine the maximum potential of BitX.

Speedup	2.61x~4.82x (fp32), 2.00x (16 fixed point), 4.98x and 14.76x higher over the baseline based on software pruning
Accuracy	Negligible accuracy under fp32, about 1% accuracy improvement under 16-bit fixed point
Accelerator Performance	2.00x and 3.79x performance improvement compared with other SOTA accelerators

BitX **is designed for** ***flexible*** **and** ***versatile*** **DNN inference for the most tasks**

Motivation

weight sparsity versus bit sparsity

Model	Weight Sparity	Bit Sparity
DenseNet121	4.84%	48.64%
ResNet50	0.33%	48.64%
ResNet152	0.75%	48.64%
ResNext50_32x4d	0.37%	48.64%
ResNext101_32x8d	3.43%	48.65%
InceptionV3	0.05%	48.64%
MNASNet0.5	0.00%	48.60%
MNASNet1.0	8.07%	48.98%
MobileNetV2	0.01%	48.67%
ShuffleNetV2_x0_5	0.00%	48.36%
ShuffleNetV2_x1_0	1.53%	48.63%
SqueezeNet1_0	0.05%	48.64%
SqueezeNet1_1	0.02%	48.64%

Very limited headroom

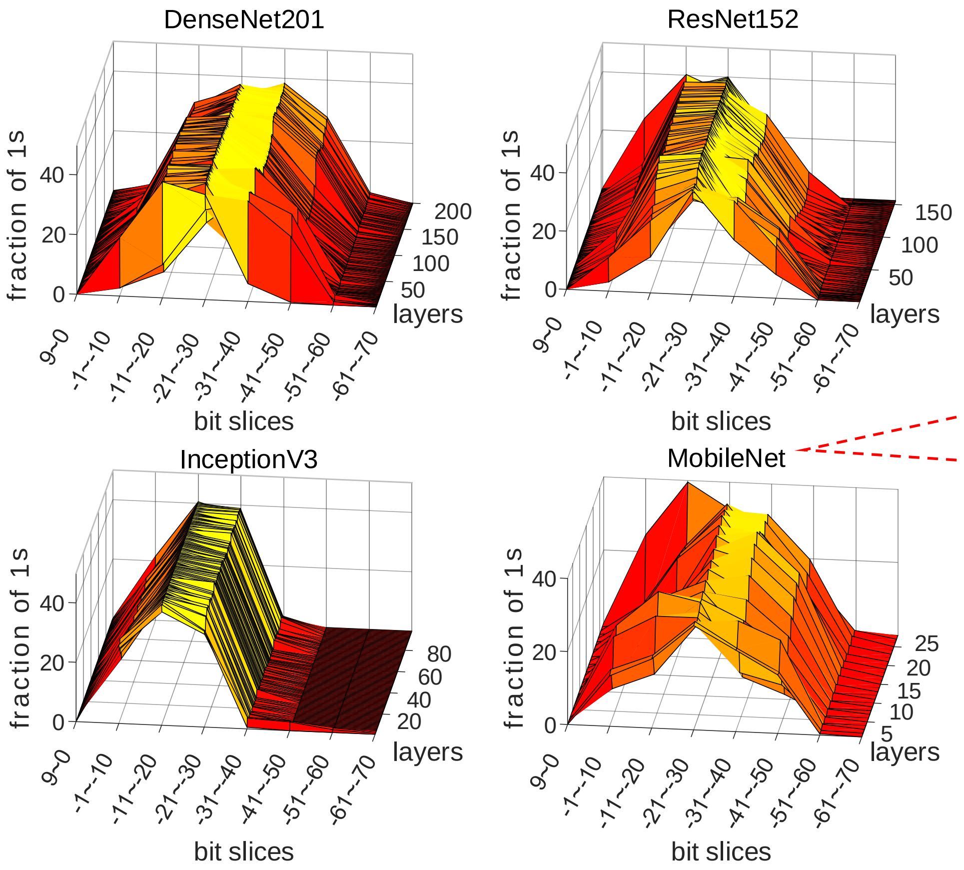
Weight sparsity : the values below 10^{-5} over the total parameter size

Significantly abundant

Bit sparsity : total bit 0s over the total "bit count" of the mantissas

Motivation

Trivial bits



Distribution analysis of bit 1s

X-axis : the bit slice of the binary represented weight (in fp32)
Y-axis : the fraction of bit '1'
All the evaluated DNNs exhibit an "arched" shape.

The central bit slices own most of the bit 1s (~ 40%). While all these bits are tiny. Taking bit significance 2^{-21} ~ 2^{-30} as the representative, the equivalent decimals are in range: 0.000000477 (~10^{-8}) to 0.000000000931 (~10^{-11})

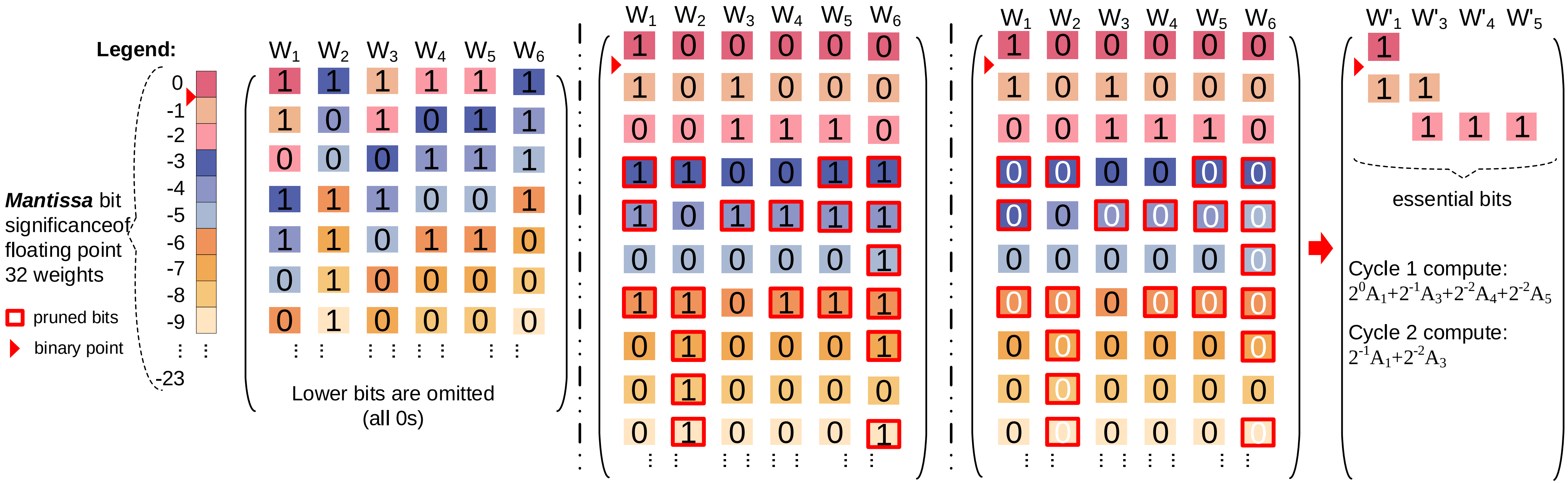
Problem tackled in this work

Goal：pinpoint the essential bits and prune away the useless bits

Genetic 0 bits	Zero-skipping mechanism to avoid the ineffectual computations caused by the zero bits -- **Easy to implement ^_^**
Trivial 1 bits	The impact of a single bit to the whole network is not that easy to be determined -- **How to tackle this problem? -_-!**

We intend to solve the two issues in BitX!

Methodology – core concept



(a) original weight matrix (b) exponents alignment (c) bit slice extraction

Weights represented in floating-point 32 mode.

Different colors indicate the bit significance from 2^{-1} to 2^{-9} after the binary point

Methodology – bit pruning

Approximating Matrix Multiplication

- **Given an m×n matrix A and an n × p matrix B,**
- **The product AB, is equivalent to the sum of n rank-one matrices**
- $AB = \sum_{i=1}^{n} \left(A^{(i)}\right) \left(\quad B_{(i)} \quad \right)$
  - ➢ **$A^{(i)}$ the i-th column of A**
  - ➢ **$B_{(i)}$ the i-th row of B**
  - ➢ **Each term in the summation is a rank-one matrix**

Metric of selecting rank-one matrices

Column vector $[A_1, A_2 \ldots A_j \ldots A_n]^T$ of activation matrix

Row vector of bit weight matrix

$$p_i = \frac{|A^{(i)}||W_{(i)}|}{\sum_{i'=1}^{l} |A^{(i')}||W_{(i')}|}$$

Exponent at position j of i-th row

$$= \frac{|A^{(i)}| \times \sqrt{\sum_{j=1}^{n} \left(2^{E_i^j} \times v_j\right)^2}}{\sum_{i'=1}^{l} \left(|A^{(i')}| \times \sqrt{\sum_{j=1}^{n} \left(2^{E_{i'}^j} \times v_j\right)^2} \right)}$$

j-th bit of the i-th row vector in W

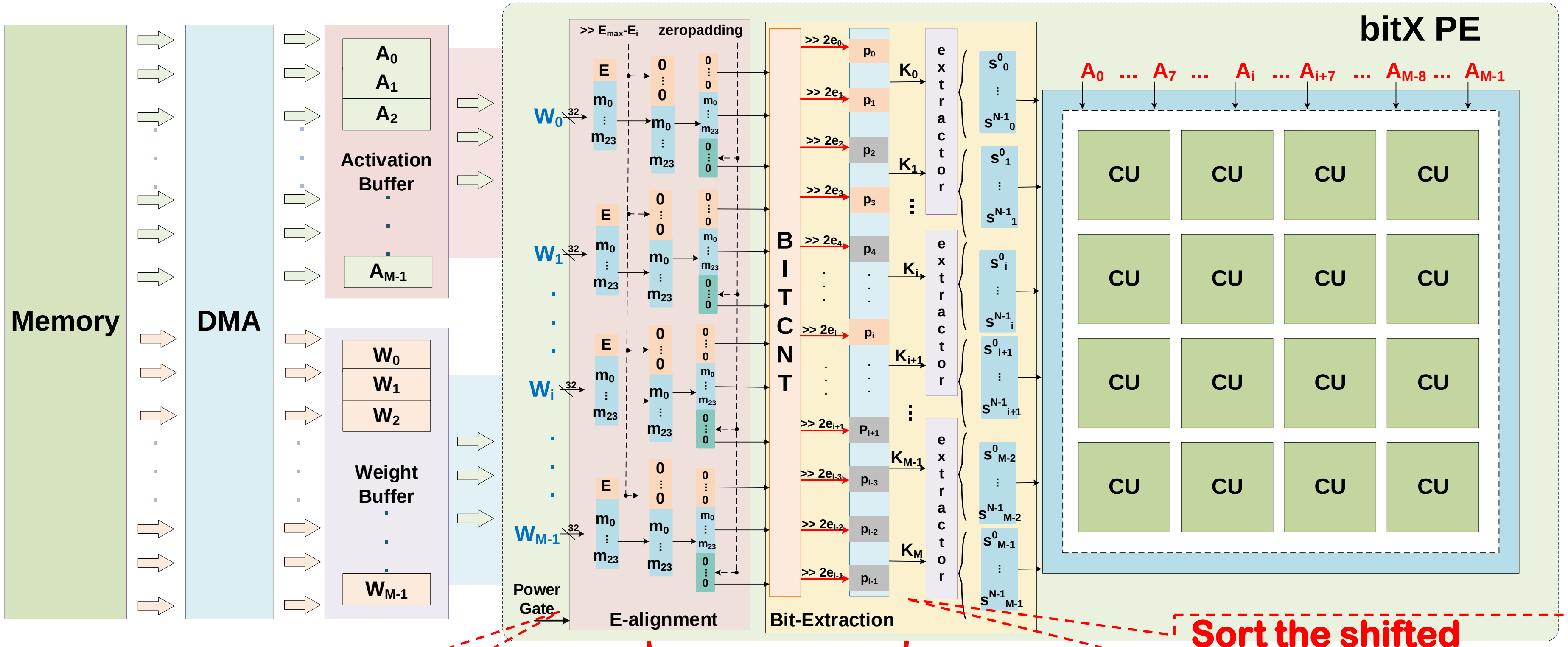
$$= \frac{\sqrt{(2^{E_i})^2 \times BitCnt(i)}}{\sum_{i'=1}^{l} \sqrt{\left(2^{E_{i'}}\right)^2 \times BitCnt(i')}}$$

A constant

Two determining factors: E_i and $BitCnt(i)$

Methodology – BitX accelerator

Data shifter Zero-padding

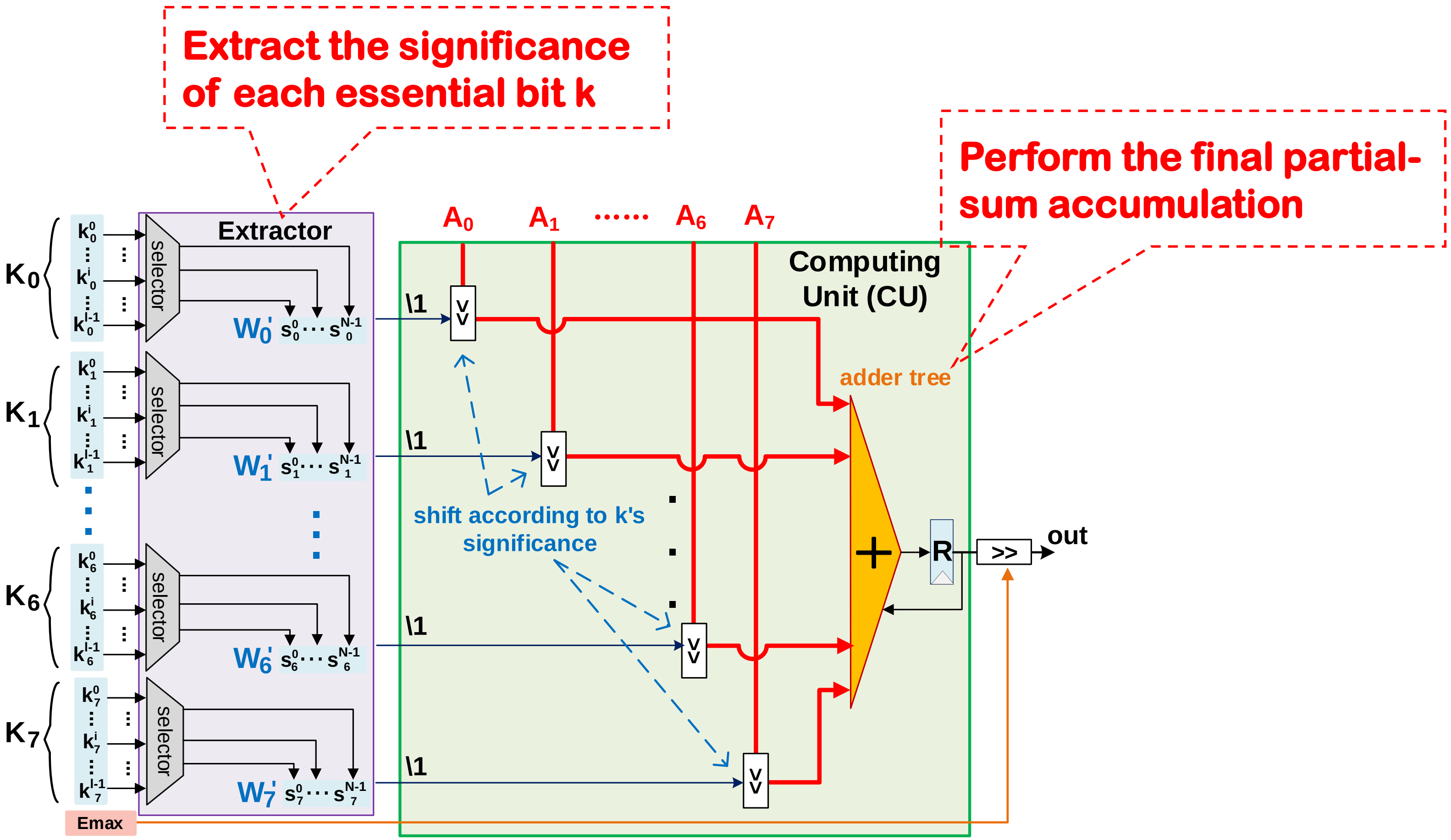


Bypassed for fixed-point DNN

Perform bit pruning algorithm

Sort the shifted BitCnt(i) **and select the top n largest rows**

Methodology – computing units



Evaluation

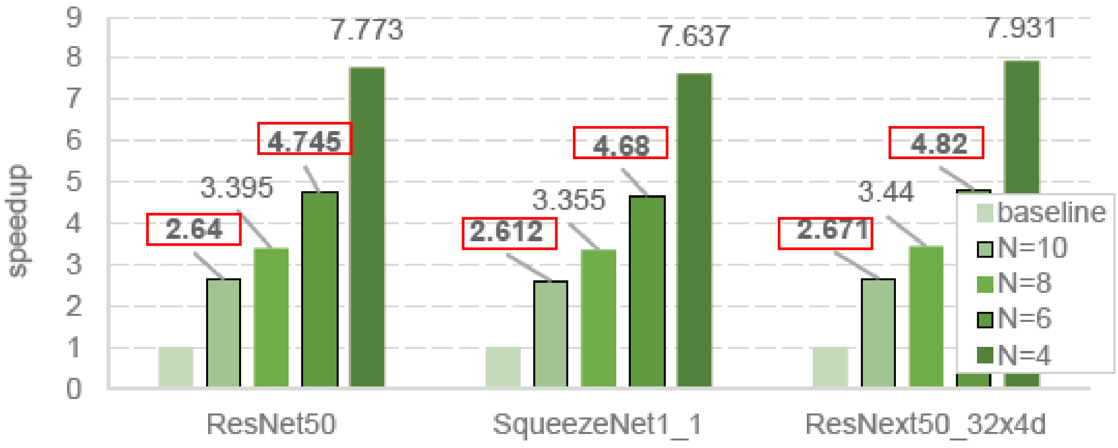
- **Accuracy & Sparsity**

Parameter to control the granularity of pruning

Model	Original	N=10	N=8	N=6	N=4
DenseNet121	71.96/1x	71.95/1.34x	71.00/1.47x	71.00/1.62x	65.00/1.76x
DenseNet161	75.28/1x	75.20/1.32x	75.14/1.46x	74.79/1.61x	72.00/1.76x
DenseNet169	73.75/1x	73.56/1.31x	73.55/1.45x	73.55/1.60x	68.62/1.75x
Densenet201	74.56/1x	74.46/1.30x	74.40/1.44x	74.24/1.59x	69.00/1.74x
ResNet18	67.28/1x	67.09/1.64x	67.00/1.73x	66.72/1.81x	62.52/1.90x
ResNet34	71.32/1x	71.11/1.65x	71.10/1.73x	70.92/1.82x	68.00/1.90x
ResNet50	74.50/1x	74.50/1.41x	74.51/1.54x	74.10/1.67x	67.00/1.80x
ResNet101	76.00/1x	76.06/1.43x	76.05/1.55x	75.76/1.68x	69.02/1.81x
ResNet152	77.02/1x	76.56/1.44x	76.55/1.56x	76.46/1.69x	72.30/1.81x
ResNext50_32x4d	76.29/1x	75.99/1.24x	75.96/1.39x	75.67/1.56x	65.01/1.72x
ResNext101_32x8d	78.24/1x	78.20/1.27x	78.30/1.42x	78.10/1.58x	73.00/1.74x
SqueezeNet1_1	54.84/1x	54.86/1.42x	54.70/1.54x	54.40/1.67x	47.30/1.80x
Avg. loss / sparsity	**0.000/1x**	**0.131/1.40x**	**0.242/1.52x**	**0.444/1.66x**	**6.023/1.79x**

1. less than 0.5% average accuracy loss at N = 10, 8, 6.
2. Accuracy improvement on some models.

- **Speedup**



BitX exhibits promising speedup of ~2.6x at N = 10, and ~4.8x at N = 6.

INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING

acm In-Cooperation sighpc

Evaluation

- **Design Space Exploration**

ResNet50	Original Accuracy: 74.50				DenseNet121	Original Accuracy: 71.96			
	N=10	N=8	N=6	N=4		N=10	N=8	N=6	N=4
M=8	74.50	74.51	74.10	67.00	M=8	71.95	71.00	71.00	65.00
M=16	74.54	74.40	73.60	61.00	M=16	71.97	72.00	71.00	62.00
M=32	74.00	74.50	73.50	58.20	M=32	72.03	72.00	71.00	58.00
M=64	74.39	74.00	73.00	53.30	M=64	71.00	71.00	70.00	55.00
M=128	74.41	74.32	72.70	46.30	M=128	71.00	71.00	70.00	49.20
M=256	74.51	74.40	72.80	46.80	M=256	71.84	71.00	69.00	49.00
M=512	74.30	74.26	71.90	39.40	M=512	71.83	71.60	69.00	34.00
ResNext101_32x8d	Original Accuracy: 78.24				SqueezeNet1_1	Original Accuracy: 54.84			
	N=10	N=8	N=6	N=4		N=10	N=8	N=6	N=4
M=8	78.20	78.30	78.10	73.00	M=8	54.86	54.70	54.40	47.30
M=16	78.20	78.00	77.50	66.00	M=16	54.80	54.74	53.00	41.60
M=32	78.20	78.00	78.20	65.00	M=32	54.00	54.50	53.64	41.70
M=64	78.20	78.20	77.30	62.00	M=64	54.70	54.77	53.50	37.10
M=128	78.20	78.10	77.30	57.00	M=128	54.40	54.48	52.80	34.66
M=256	78.20	78.20	77.20	49.00	M=256	54.62	54.40	52.11	32.10
M=512	78.20	78.20	77.20	49.00	M=512	54.81	54.72	52.60	32.00

M:number of input weights that the accelerator could simultaneously prune

Two BitX instances:

BitX-mild (N=10, M=8)
BitX-wild (N=6, M=8)

1. M barely influences the over all accuracy scaling from 8~512 for all 4 DNNs.
2. Accuracy improvement in some models.

Evaluation

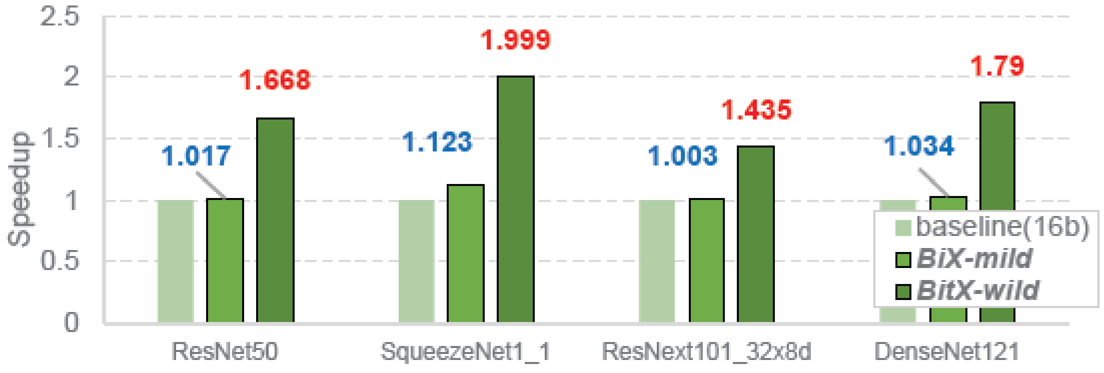
- **Performance of the Fixed-point DNN**

Accuracy

Model	Baseline(16b)	*BitX-mild*	*BitX-wild*
ResNet50	74.50	74.50 (0.00)	74.10 (-0.40)
SqueezeNet1_1	54.86	54.80 (-0.06)	54.40 (-0.46)
DenseNet121	71.00	71.90 (+0.90)	71.80 (+0.80)
ResNext101_32x8d	78.00	78.20 (+0.20)	78.10 (+0.10)

BitX-mild and *BitX-wild* both exhibit higher accuracy than most of non-pruned models.

Speedup



1. ~2x speedup in *BitX-wild.*
2. ~10% but abundant speedup in *BitX-mild.*

Evaluation

- **Working with Software-based Pruning**

BitX is orthogonal to any software-based pruning schemes

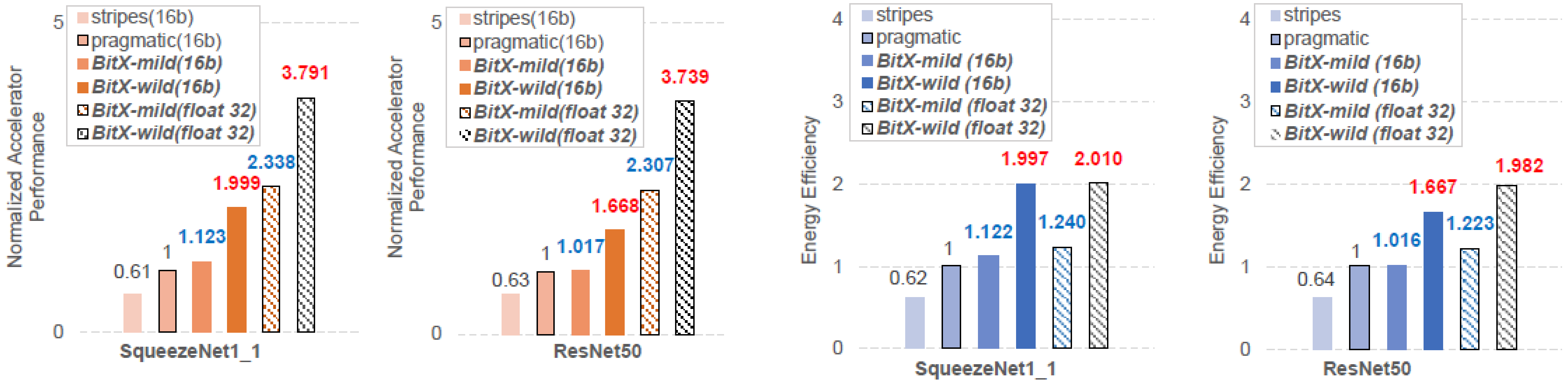
Method	mAP(%)	Speedup(x)
YoloV3 **(baseline)**	50.36	1
YoloV3 + ***BitX-mild***	**(50.42) (+0.06)**	2.75
YoloV3 + ***BitX-wild***	50.05 (-0.31)	**4.98**
YoloV3 + Slimming **(baseline)**	50.23 (-0.13)	2.35
YoloV3 + Slimming + ***BitX-mild***	**50.30 (+0.07)**	7.22
YoloV3 + Slimming + ***BitX-wild***	48.72 (-1.64)	**14.76**

Higher speedup than software based pruning

Considerable speedup than genetic model

Evaluation

- **Comparison with SOTA Accelerators**



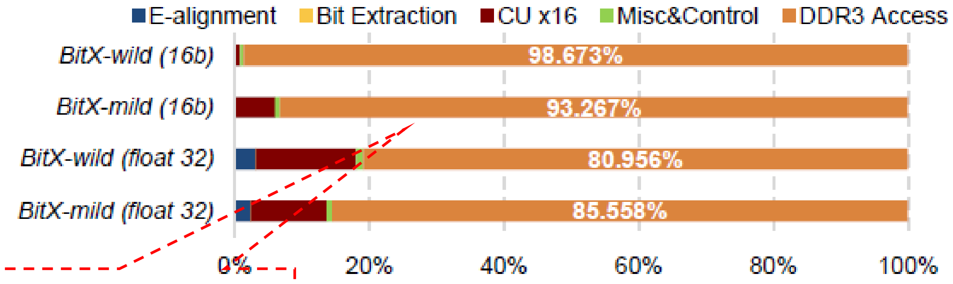
1. The speedup shows better result over other SOTA accelerator both in *BitX-wild* and *BitX-mild*.

2. The floating-point results are higher than the fixed-point results.

Energy efficiency of *BitX* also outperforms other accelerators.

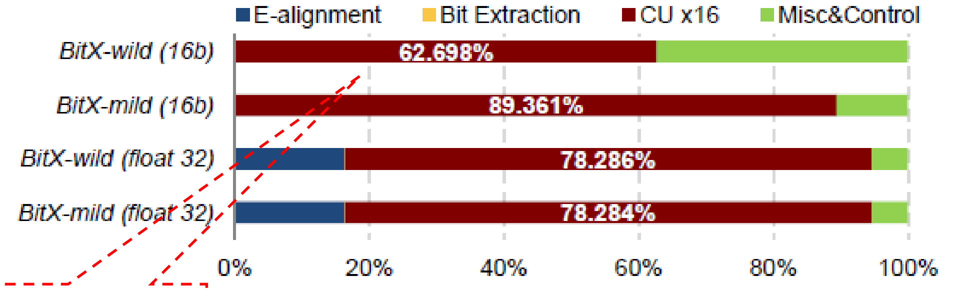
Evaluation

- **Energy breakdown**



Memory accesses dominates

(a) Full system energy breakdown



CU dominates

(b) BitX PE-only energy breakdown

- **Area and Power breakdown**

Precision	*BitX* (floating-point 32)		*BitX* (16b fixed point)
Item	Area (mm^2)	Power (mW)	Power (mW)
E-alignment	0.017 (43.60%)	11.15 (16.20%)	–
Bit Extraction	0.008 (20.10%)	0.04 (0.05%)	0.026 (0.07%)
16 CUs	0.003 (7.70%)	53.71 (78.30%)	35.81 (98.40%)
Misc&Control	0.011 (28.20%)	3.72 (5.40%)	0.576 (1.60%)
Total	0.039	68.62	36.41

1. Area : only 0.039 mm^2
2. 36.41 mW : high speedup, high power consumption
3. 68.62 mW : low speedup, low power consumption

Recap

The contribution of this work

① Propose a novel hardware runtime pruning method -- BitX, to empower versatile DNN inference

- Software effortless — No retraining! No fine-tuning!
- Orthogonal to the existing software pruning methodologies — obtain additional speedup
- Multi-precision support — floating point & fixed point DNNs

② Propose a deep learning accelerator capable of unprecedented hardware runtime pruning to mine the maximum potential of BitX.

Applications, and what's more?

Thanks for listening!

Questions?

Hongyan Li[1,2], Hang Lu[1,2], Jiawen Huang[1], Wenxu Wang[1,2], Mingzhe Zhang[1], Wei Chen[1], Liang Chang[3] , Xiaowei Li[1,2]

[1]State Key Laboratory of Computer Architecture, Institute of Computing Technology, CAS, Beijing, China

[2]University of Chinese Academy of Sciences, Beijing, China

[3]University of Electronic Science and Technology of China, Chengdu, China