CMSC 5743

Efficient Computing of Deep Neural Networks

Implementation 03: GEMM-3

Bei Yu
CSE Department, CUHK
byu@cse.cuhk.edu.hk

(Latest update: March 2, 2023)

Spring 2023

Overview

1. Sparse Convolution
2. Sparse Hardware Architecture

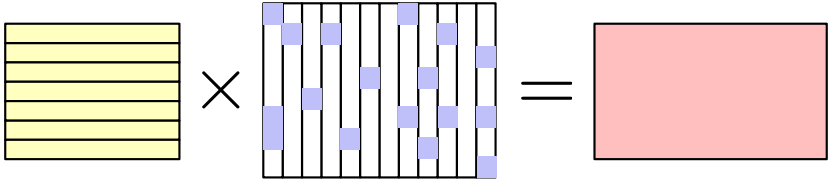
Overview

1. Sparse Convolution
2. Sparse Hardware Architecture

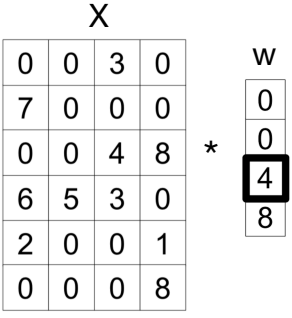
Sparse Convolution

Sparse Convolution

- Our DNN may be redundant, and sometimes the filters may be sparse
- Sparsity can be helpful to overcome over-fitting



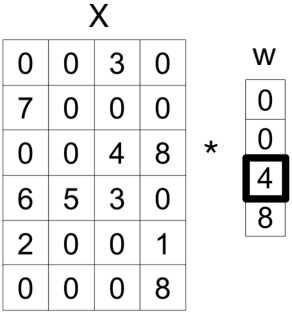
Sparse Convolution: Naive Implementation 1



Algorithm Sparse Convlution Naive 1

1: **for all** w[i] **do**
2: **if** w[i] = 0 **then**
3: Continue;
4: **end if**
5: output feature map $\boldsymbol{Y} \leftarrow \boldsymbol{X} \times w$[i];
6: **end for**

Sparse Convolution: Naive Implementation 1



Algorithm Sparse Convlution Naive 1

1: **for all** w[i] **do**
2: **if** w[i] = 0 **then**
3: Continue;
4: **end if**
5: output feature map $\boldsymbol{Y} \leftarrow \boldsymbol{X} \times w$[i];
6: **end for**

BAD implementation for Pipeline!

Instr. No.	Pipeline Stage						
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	1	2	3	4	5	6	7

Sparse Matrix Representation

A

0	0	3	0
7	0	0	0
0	0	4	8
6	5	3	0
2	0	0	1
0	0	0	8

A matrix example

rowptr

- row0 (3,2)
- row1 (7,0)
- row2 (4,2), (8,3)
- row3 (6,0), (5,1), (3,2)
- row4 (2,0), (1,3)
- row5 (8,3)

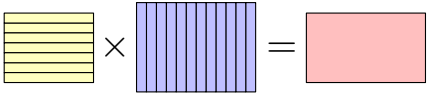
Compressed Sparse Row (CSR)

colptr

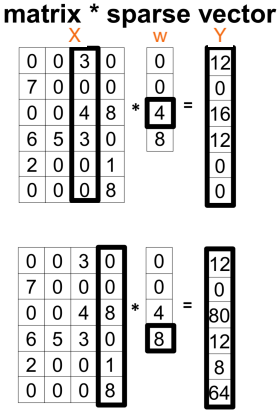
- col0 (7,1), (6,3), (2,4)
- col1 (5,3)
- col2 (3,0), (4,2), (3,3)
- col3 (8,2), (1,4), (8,5)

Compressed Sparse Column (CSC)

- CSR: Good for operation on feature maps
- CSC: Good for operation on filters
- We have better control on filters, thus usually CSC.



Sparse Convolution: Naive Implementation 2



- BAD implementation for Spatial Locality!
- Poor memory access patterns

SOTA 2: Sparse Convolution

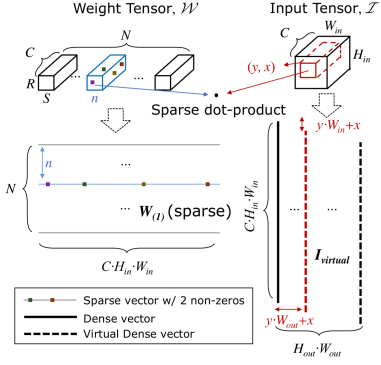


Figure 1: Conceptual view of the direct sparse convolution algorithm. Computation of output value at (y,x)th position of nth output channel is highlighted.

```
for each output channel n {
 for j in [W.rowptr[n], W.rowptr[n+1]) {
  off = W.colidx[j]; coeff = W.value[j]
  for (int y = 0; y < H_OUT; ++y) {
   for (int x = 0; x < W_OUT; ++x) {
    out[n][y][x] += coeff*in[off+f(0,y,x)]
   }
  }
 }
}
```

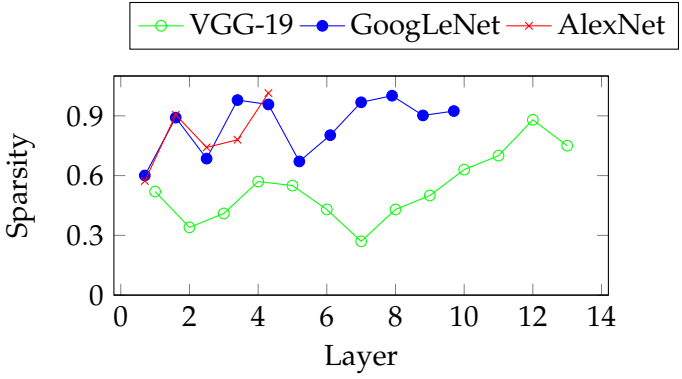
Figure 2: Sparse convolution pseudo code. Matrix **W** has *compressed sparse row* (CSR) format, where `rowptr[n]` points to the first non-zero weight of nth output channel. For the jth non-zero weight at (n,c,r,s), `W.colidx[j]` contains the offset to (c,r,s)th element of tensor `in`, which is pre-computed by layout function as $f(c,r,s)$. If `in` has CHW format, $f(c,r,s) = (cH_{in}+r)W_{in}+s$. The "virtual" dense matrix is formed on-the-fly by shifting `in` by $(0,y,x)$.

1

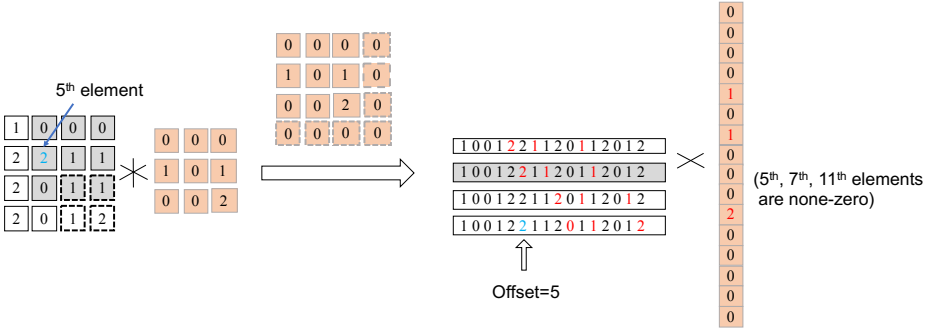
[1] Jongsoo Park et al. (2017). "Faster CNNs with direct sparse convolutions and guided pruning". In: *Proc. ICLR*.

Discussion: Sparse-Sparse Convolution

- Sparsity is a desired property for computation acceleration. (`cuSPARSE` library, direct sparse convolution, etc.)
- Sometimes not only the filters but also the input feature maps are sparse.



Discussion: Sparse-Sparse Convolution



- Efficient programming implementation required; (Improve pipeline efficiency)
- When sparsity$(input) = 0.9$, sparsity$(weight) = 0.8$, more than $10\times$ speedup;
- Some other issues:
  - How to be compatible with pooling layer?
  - Transform between dense & sparse formats

Overview

1. Sparse Convolution
2. Sparse Hardware Architecture

Sparse Hardware Architecture

EIE: Efficient Inference Engine on Compressed Deep Neural Network

Han et al.
ISCA 2016

Deep Learning Accelerators

- First Wave: Compute (Neu Flow)
- Second Wave: Memory (Diannao family)
- Third Wave: Algorithm / Hardware Co-Design (EIE)

Google TPU: “This unit is designed for dense matrices. Sparse architectural support was omitted for time-to-deploy reasons. Sparsity will have high priority in future designs”

[Han et al. ISCA'16]

EIE: the First DNN Accelerator for Sparse, Compressed Model

0 * A = 0

Sparse Weight

90% *static* sparsity

W * 0 = 0

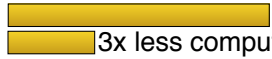
Sparse Activation

70% *dynamic* sparsity

~~2.09~~, ~~1.92~~=> 2

Weight Sharing

4-bit weights



[Han et al. ISCA'16]

EIE: Parallelization on Sparsity

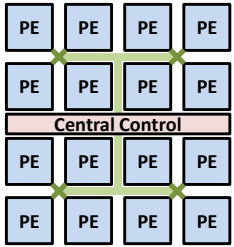
$$\vec{a}\ \begin{pmatrix} 0 & \boldsymbol{a_1} & 0 & a_3 \end{pmatrix}$$

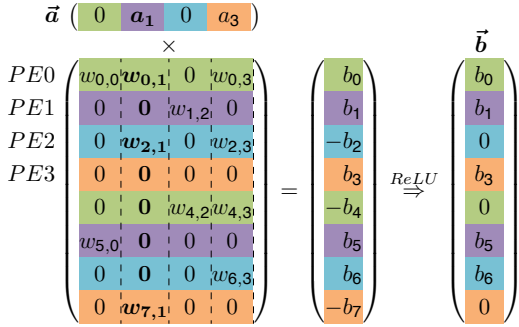
$$\times$$

$$\begin{pmatrix} w_{0,0} & \boldsymbol{w_{0,1}} & 0 & w_{0,3} \\ 0 & \boldsymbol{0} & w_{1,2} & 0 \\ 0 & \boldsymbol{w_{2,1}} & 0 & w_{2,3} \\ 0 & \boldsymbol{0} & 0 & 0 \\ 0 & \boldsymbol{0} & w_{4,2} & w_{4,3} \\ w_{5,0} & \boldsymbol{0} & 0 & 0 \\ 0 & \boldsymbol{0} & 0 & w_{6,3} \\ 0 & \boldsymbol{w_{7,1}} & 0 & 0 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ -b_2 \\ b_3 \\ -b_4 \\ b_5 \\ b_6 \\ -b_7 \end{pmatrix} \overset{ReLU}{\Rightarrow} \vec{\boldsymbol{b}} \begin{pmatrix} b_0 \\ b_1 \\ 0 \\ b_3 \\ 0 \\ b_5 \\ b_6 \\ 0 \end{pmatrix}$$

[Han et al. ISCA'16]

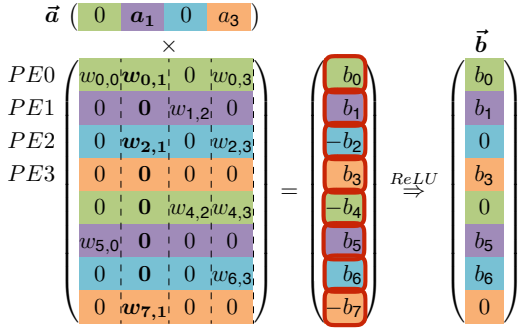
EIE: Parallelization on Sparsity





[Han et al. ISCA'16]

Dataflow

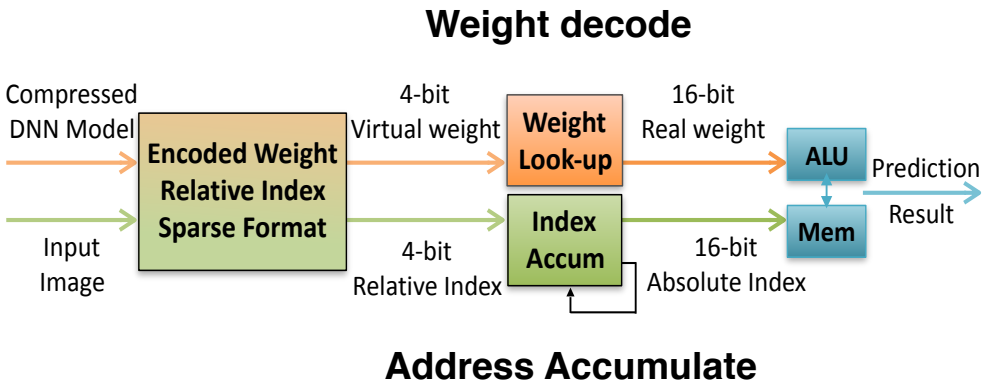


rule of thumb:

0 * A = 0 W * 0 = 0

[Han et al. ISCA'16]

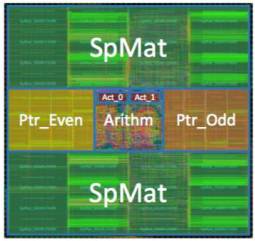
EIE Architecture



rule of thumb: 0 * A = 0 W * 0 = 0 ~~2.09, 1.92~~=> 2

[Han et al. ISCA'16]

Post Layout Result of EIE

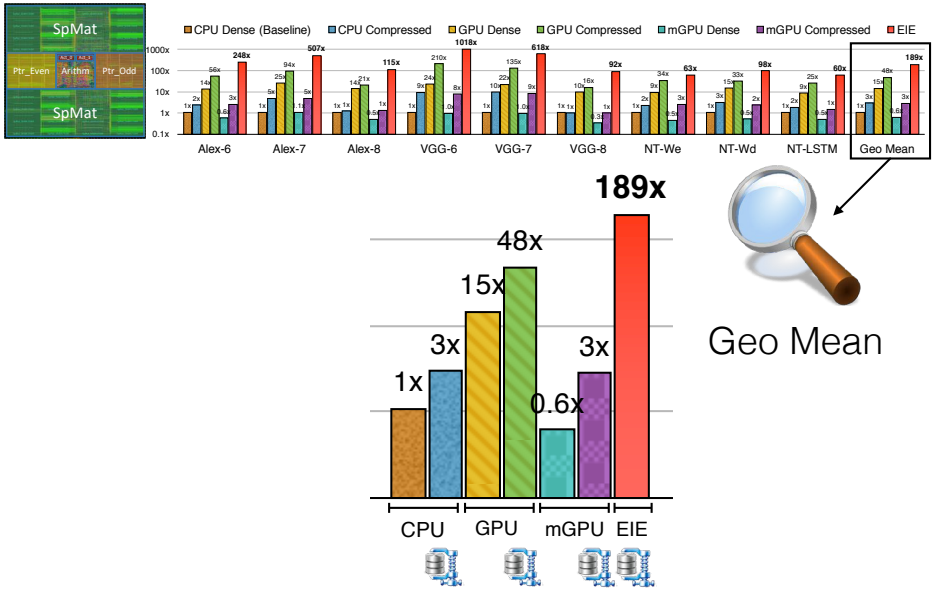


Technology	40 nm
# PEs	64
on-chip SRAM	8 MB
Max Model Size	84 Million
Static Sparsity	10x
Dynamic Sparsity	3x
Quantization	4-bit
ALU Width	16-bit
Area	40.8 mm^2
MxV Throughput	81,967 layers/s
Power	586 mW

1. Post layout result
2. Throughput measured on AlexNet FC-7

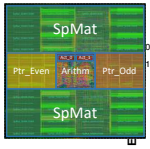
[Han et al. ISCA'16]

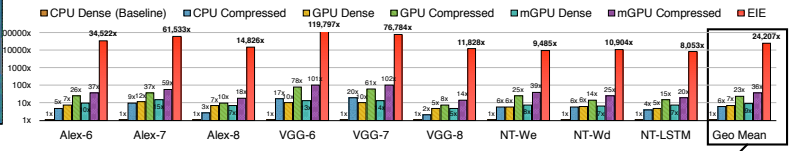
Speedup on EIE

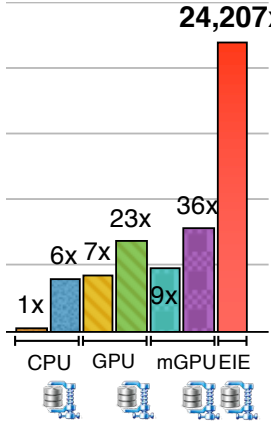


[Han et al. ISCA'16]

Energy Efficiency on EIE



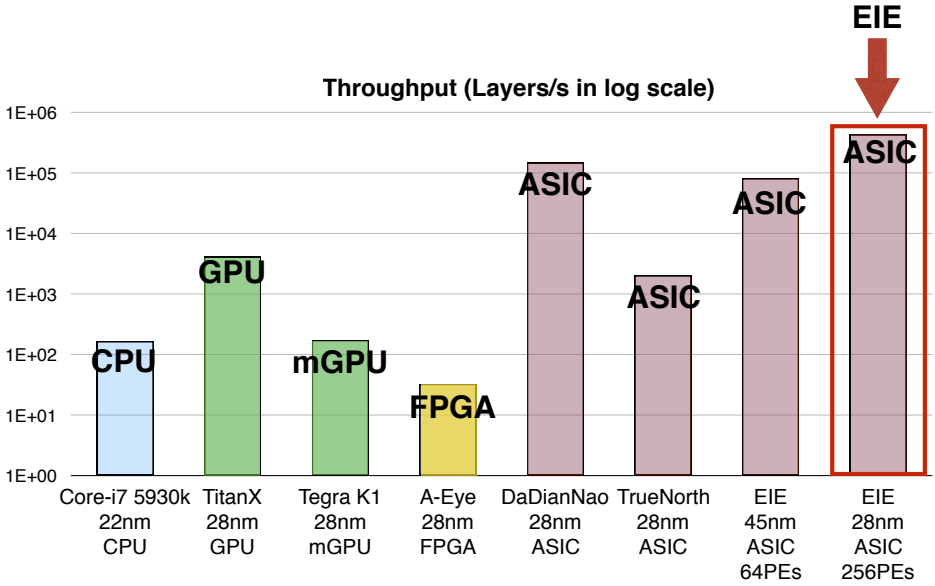




Geo Mean

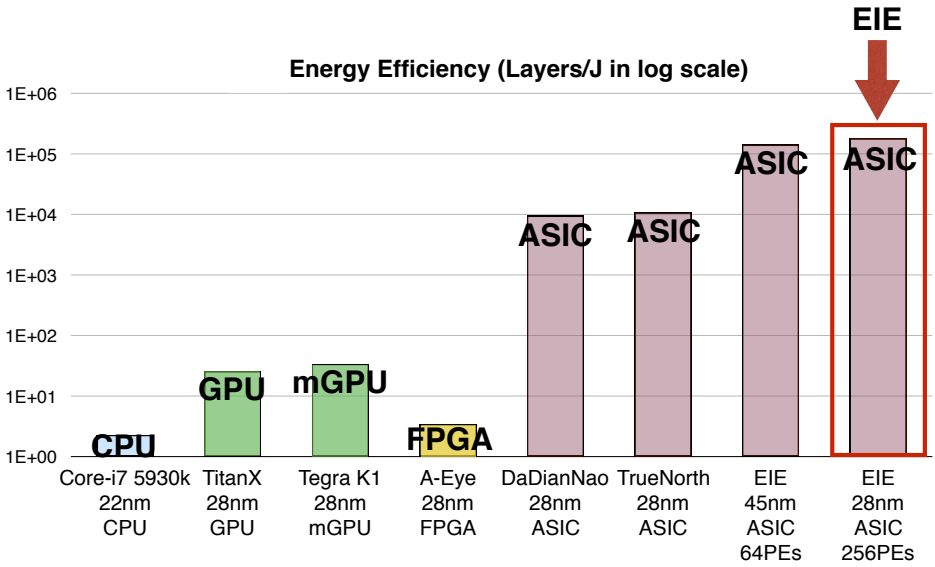
[Han et al. ISCA'16]

Comparison: Throughput

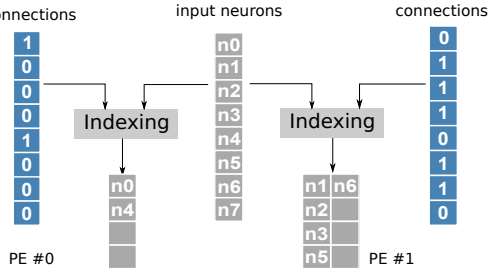


[Han et al. ISCA'16]

Comparison: Energy Efficiency



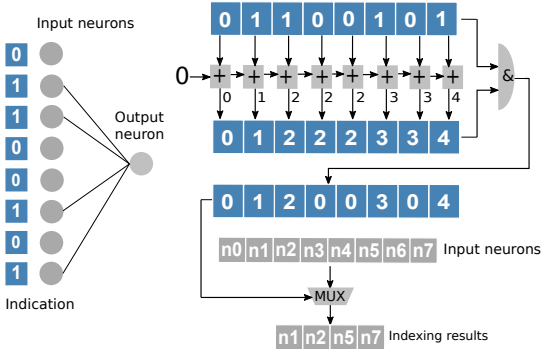
Indexing Module (IM) for sparse data



- IM is used for indexing needed neurons of sparse networks with different levels of sparsities.
- A centralized IM is designed in the buffer controller and only transfer the indexed neurons to processing engines.

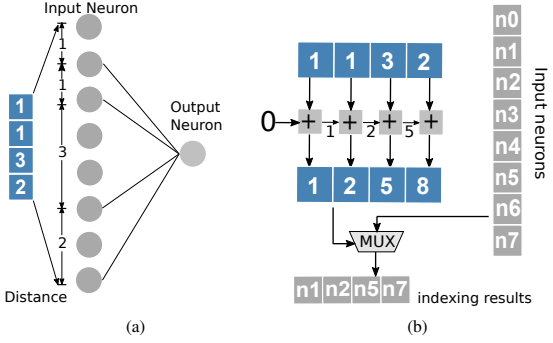
[2]Shijin Zhang et al. (2016). "Cambricon-x: An accelerator for sparse neural networks". In: *Proc. MICRO*. IEEE, pp. 1–12.

Direct indexing and hardware implementation



- Neurons are selected from all input neurons directly based on existed connections in the binary string.

Step indexing and hardware implementation

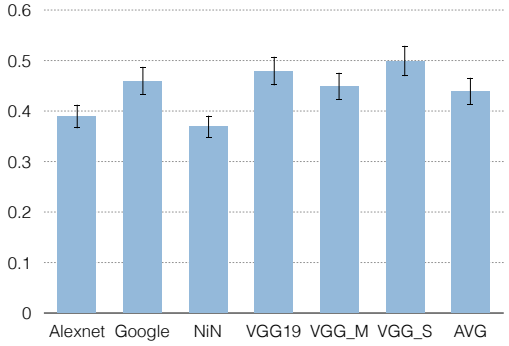


- Neurons are selected based on the distances between input neurons with existed synapses.

Feature Sparsity[3]

Lots of Runtime Zeroes

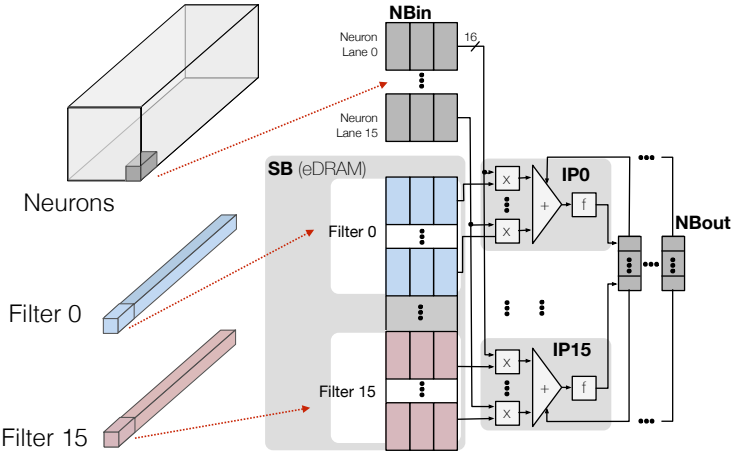
Ineffectual zero computations.



Fraction of zero neurons in multiplications

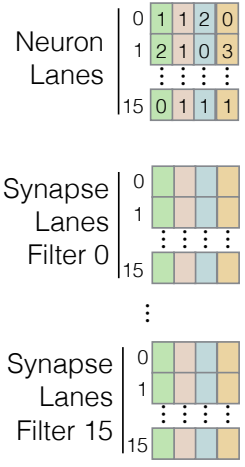
[3]Jorge Albericio et al. (2016). "Cnvlutin: Ineffectual-neuron-free deep neural network computing". In: *ACM SIGARCH Computer Architecture News* 44.3, pp. 1–13.

DaDianNao[4]

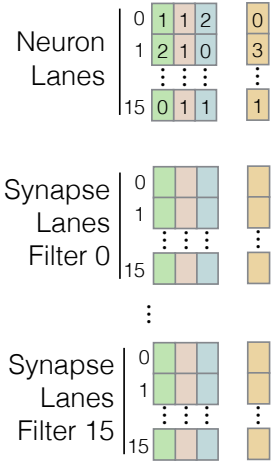


[4]Yunji Chen et al. (2014). "Dadiannao: A machine-learning supercomputer". In: *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE, pp. 609–622.

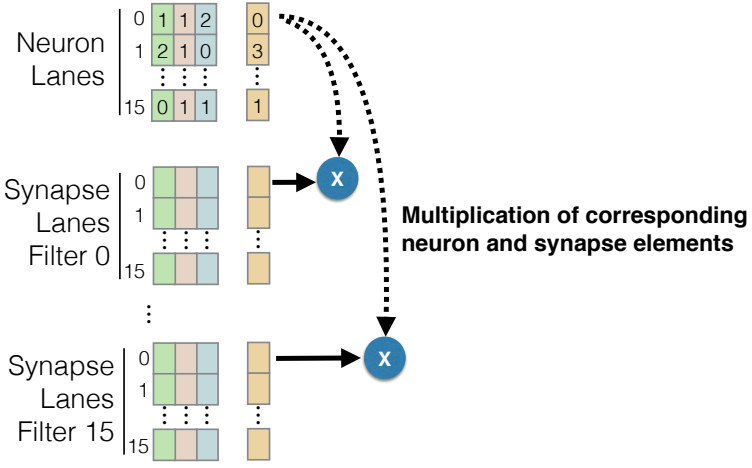
Processing in DaDianNao



Processing in DaDianNao

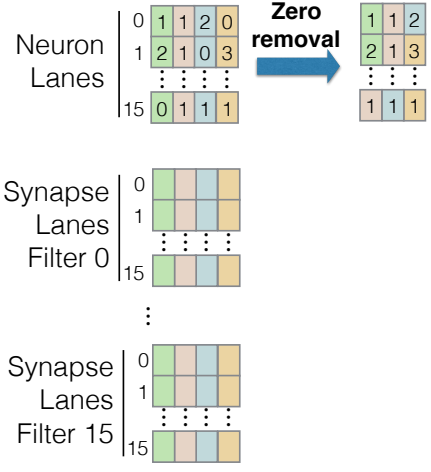


Processing in DaDianNao



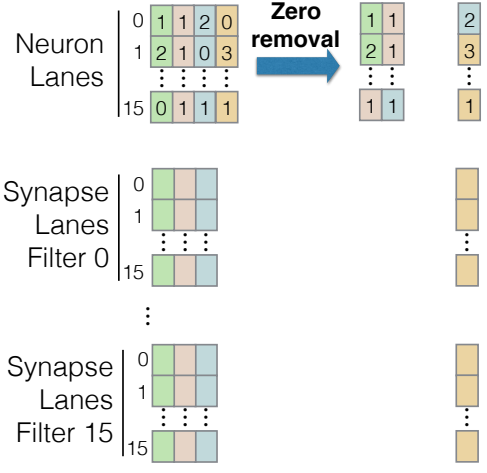
Processing in DaDianNao

Zero removal.



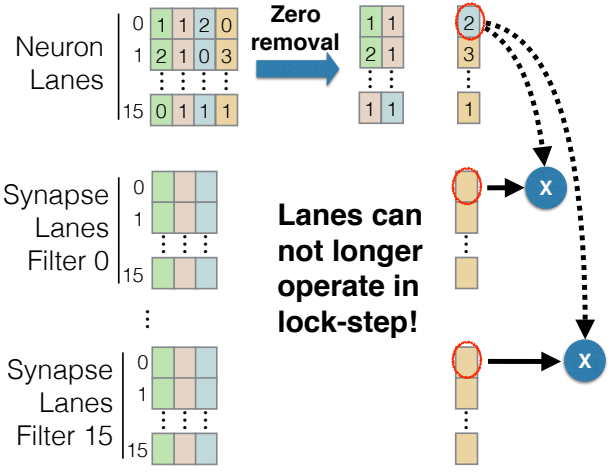
Processing in DaDianNao

Zero removal.

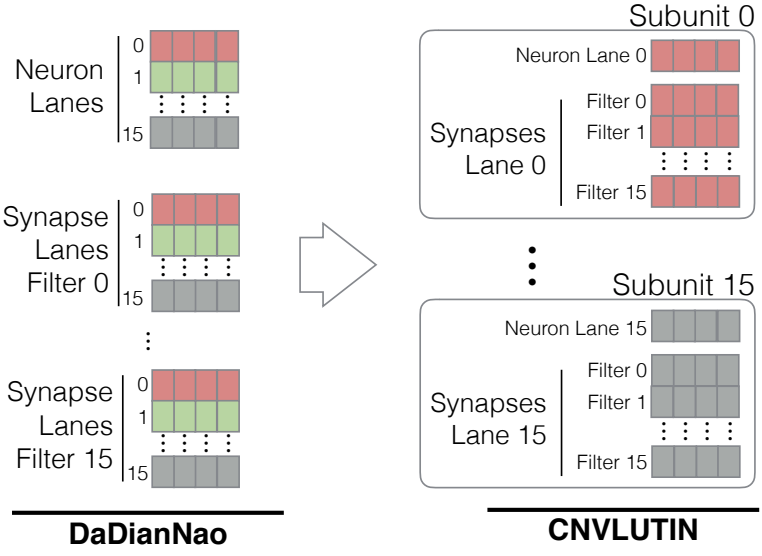


Processing in DaDianNao

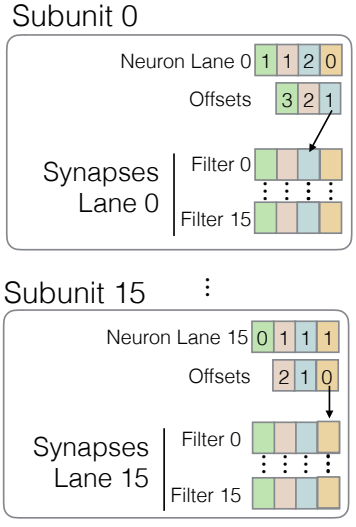
Lanes can not longer operate in lock-step.



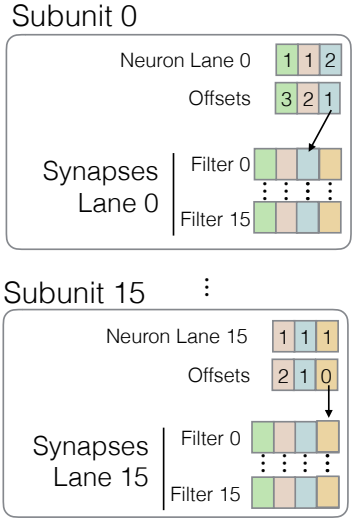
CNVLUTIN: Decoupling Lanes



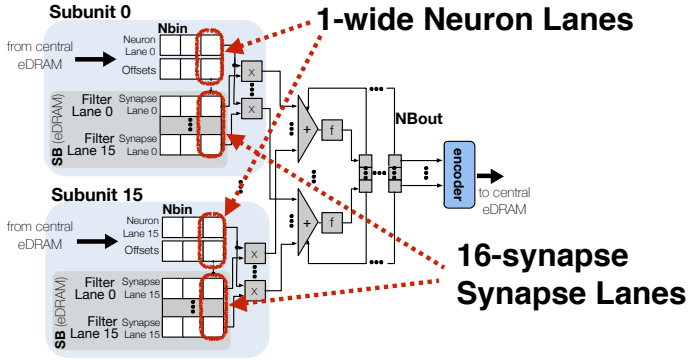
CNVLUTIN: Decoupling Lanes



CNVLUTIN: Decoupling Lanes



CNVLUTIN: Decoupling Lanes



Decoupled Neuron Lanes:
- Neuron + coordinate
- Proceed independently

Partitioned SB:
- 16-wide accesses
- 1 synapse per filter

Further Discussion: Reading List

- Wenlin Chen et al. (2015). "Compressing neural networks with the hashing trick". In: *Proc. ICML*, pp. 2285–2294
- Huizi Mao et al. (2017). "Exploring the granularity of sparsity in convolutional neural networks". In: *CVPR Workshop*, pp. 13–20
- Zhuang Liu et al. (2017). "Learning efficient convolutional networks through network slimming". In: *Proc. ICCV*, pp. 2736–2744
- Chenglong Zhao et al. (June 2019). "Variational convolutional neural network pruning". In: *Proc. CVPR*
- Junru Wu et al. (2018). "Deep k-Means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions". In: *Proc. ICML*