香港中文大學
The Chinese University of Hong Kong

CENG5030
Part 2-2: CNN Accurate Speedup

Bei Yu

(Latest update: March 25, 2019)

Spring 2019

These slides contain/adapt materials developed by

- Minsik Cho and Daniel Brand (2017). “MEC: memory-efficient convolution for deep neural network”. In: *Proc. ICML*
- Asit K. Mishra et al. (2017). “Fine-grained accelerators for sparse machine learning workloads”. In: *Proc. ASPDAC*, pp. 635–640
- Jongsoo Park et al. (2017). “Faster CNNs with direct sparse convolutions and guided pruning”. In: *Proc. ICLR*

Overview

Dense Convolution

Sparse Convolution

Reading List

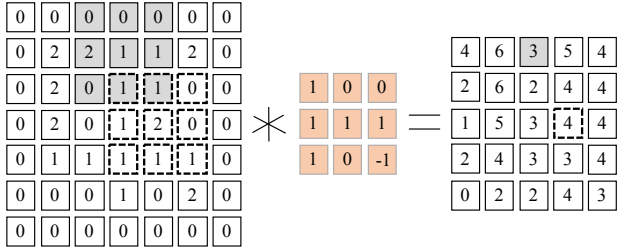
Overview

Dense Convolution

Sparse Convolution

Reading List

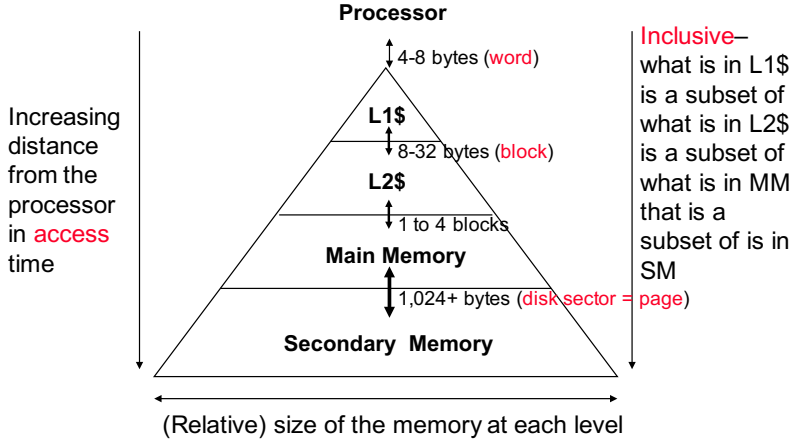
Convolution 101



Direct convolution: No extra memory overhead

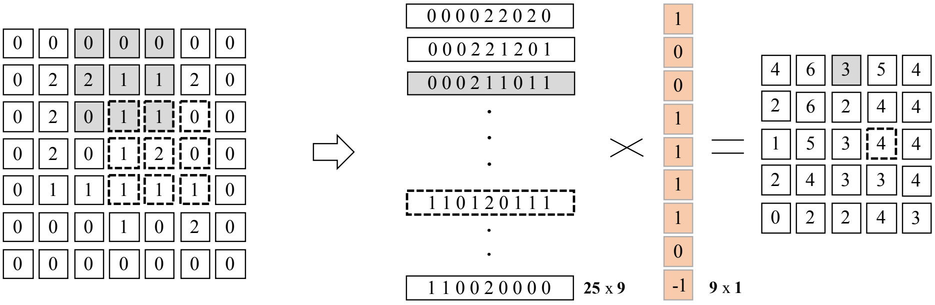
- Low performance
- Poor memory access pattern due to geometry-specific constraint
- Relatively short dot product

Background: Memory System



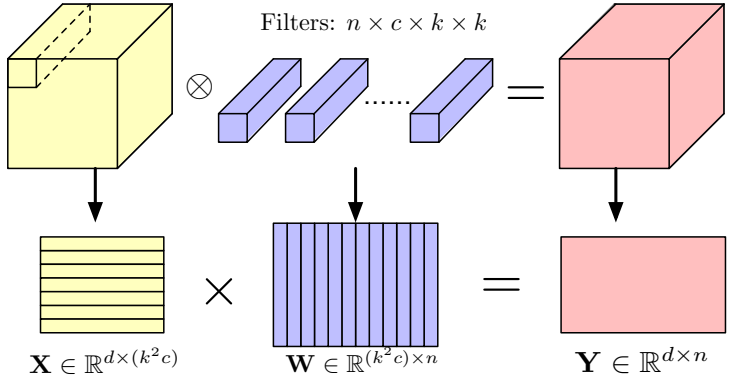
- Spatial locality
- Temporal Locality

Im2col (Image2Column) Convolution



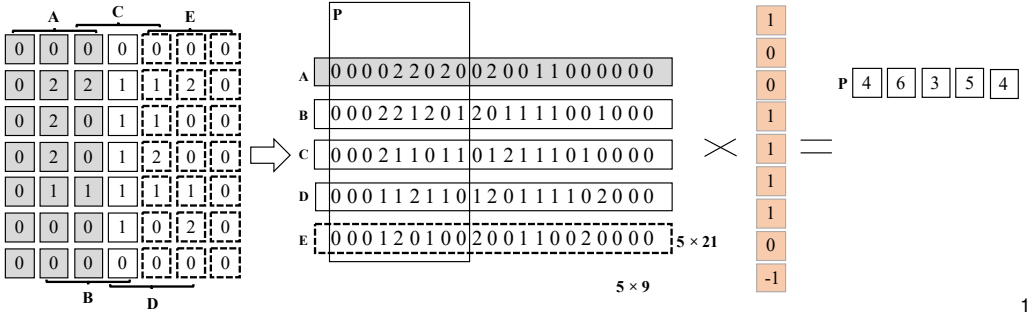
- Large extra memory overhead
- Good performance
- BLAS-friendly memory layout to enjoy SIMD/locality/parallelism
- Applicable for any convolution configuration on any platform

`Im2col` (Image2Column) Convolution



- Transform convolution to matrix multiplication
- Unified calculation for both convolution and fully-connected layers

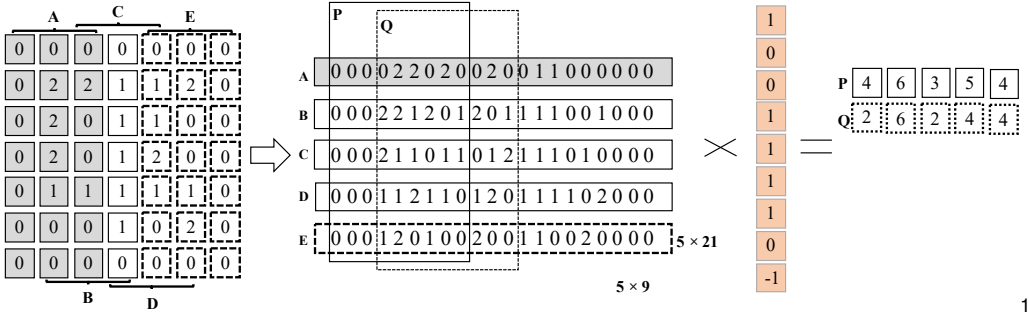
SOTA 1: Memory-efficient Convolution



- Sub matrices in the lowered matrix will be “sgemm” ed in parallel
- Smaller memory foot print, cache locality, and explicit parallelism

[1] Minsik Cho and Daniel Brand (2017). “MEC: memory-efficient convolution for deep neural network”. In: *Proc. ICML*.

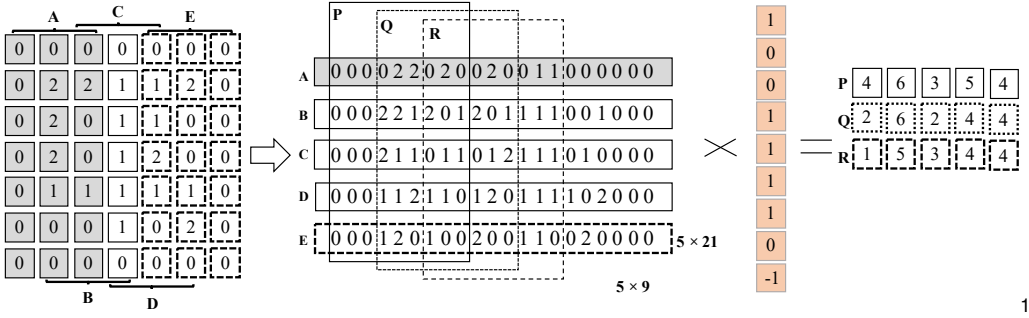
SOTA 1: Memory-efficient Convolution



- Sub matrices in the lowered matrix will be “sgemm” ed in parallel
- Smaller memory foot print, cache locality, and explicit parallelism

[1] Minsik Cho and Daniel Brand (2017). “MEC: memory-efficient convolution for deep neural network”. In: *Proc. ICML*.

SOTA 1: Memory-efficient Convolution

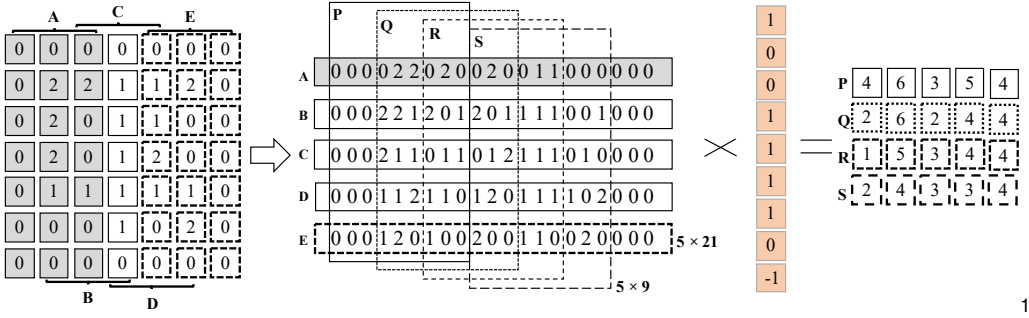


[1]

- Sub matrices in the lowered matrix will be “sgemm” ed in parallel
- Smaller memory foot print, cache locality, and explicit parallelism

[1] Minsik Cho and Daniel Brand (2017). “MEC: memory-efficient convolution for deep neural network”. In: *Proc. ICML*.

SOTA 1: Memory-efficient Convolution

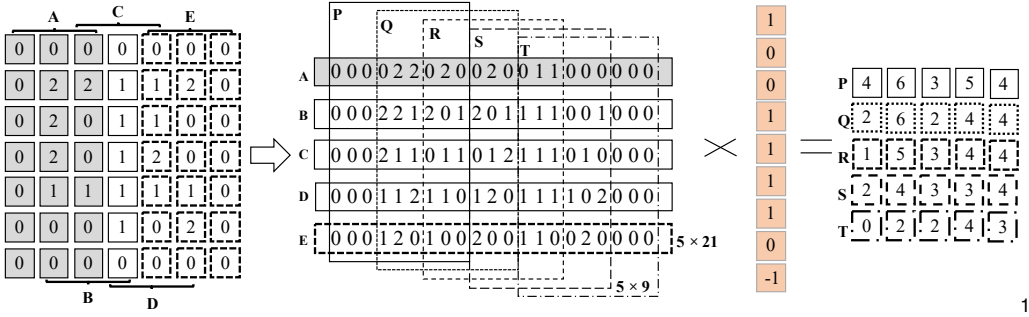


1

- Sub matrices in the lowered matrix will be “sgemm” ed in parallel
- Smaller memory foot print, cache locality, and explicit parallelism

[1] Minsik Cho and Daniel Brand (2017). “MEC: memory-efficient convolution for deep neural network”. In: *Proc. ICML*.

SOTA 1: Memory-efficient Convolution



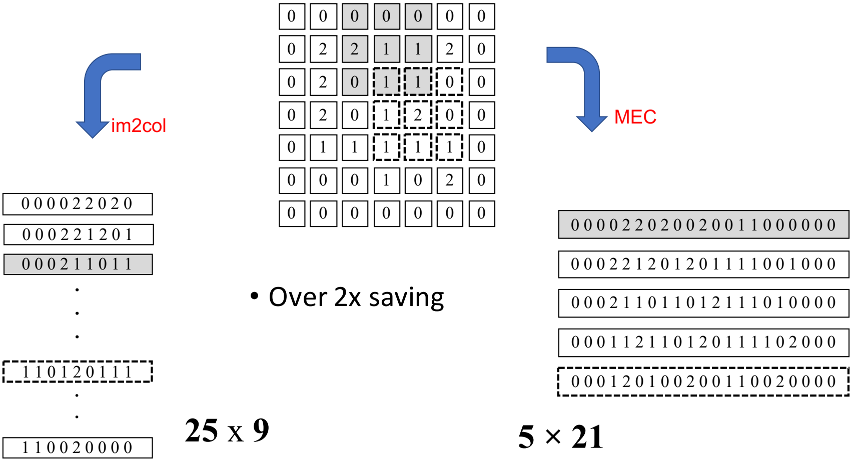
1

- Sub matrices in the lowered matrix will be “sgemm” ed in parallel
- Smaller memory foot print, cache locality, and explicit parallelism

[1] Minsik Cho and Daniel Brand (2017). “MEC: memory-efficient convolution for deep neural network”. In: *Proc. ICML*.

SOTA 1: Memory-efficient Convolution

Over $2\times$ memory saving[2]:



---

[2] Minsik Cho and Daniel Brand (2017). "MEC: memory-efficient convolution for deep neural network". In: *Proc. ICML*.

Overview

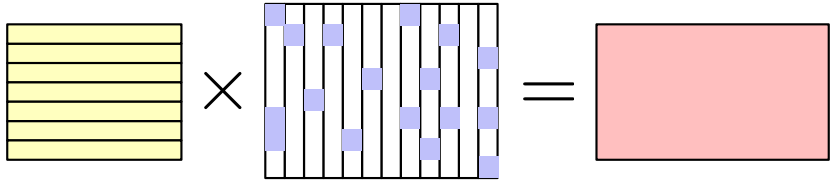
Dense Convolution

Sparse Convolution

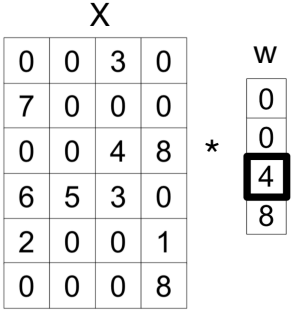
Reading List

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 1 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 $Y \leftarrow X \times w$[i];
6: **end for**

Sparse Convolution: Naive Implementation 1

X

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

*

w

0
0
4
8

Algorithm 2 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 $Y \leftarrow X \times w$[i];
6: **end for**

BAD implementation for Pipeline!

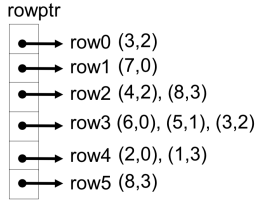
Instr. No.	Pipeline Stage						
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	1	2	3	4	5	6	7

Sparse Matrix Representation

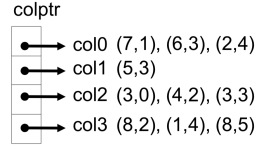
A

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

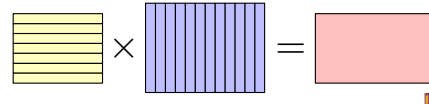


Compressed Sparse Row (CSR)



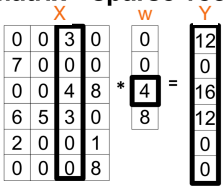
Compressed Sparse Column (CSC)

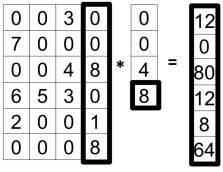
- CSR: Good for operation on feature maps
- CSC: Good for operation on filters
- We have better control on filters, thus usually CSC.



Sparse Convolution: Naive Implementation 2

matrix * sparse vector





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

SOTA 2: Sparse Convolution

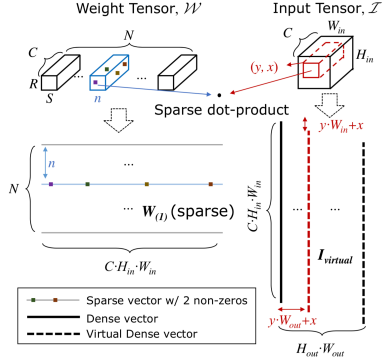


Figure 1: Conceptual view of the direct sparse convolution algorithm. Computation of output value at (y,x)th position of nth output channel is highlighted.

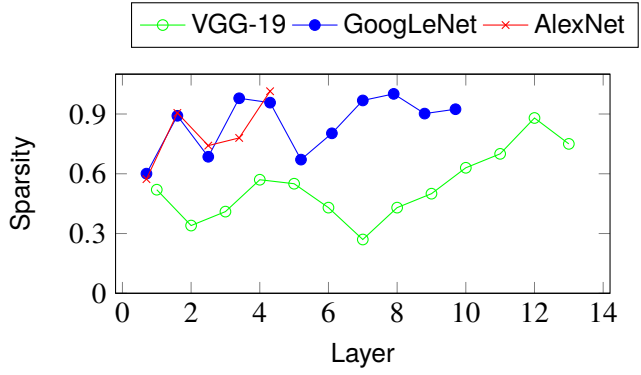
```
for each output channel n {
 for j in [W.rowptr[n], W.rowptr[n+1]) {
  off = W.colidx[j]; coeff = W.value[j]
  for (int y = 0; y < H_OUT; ++y) {
   for (int x = 0; x < W_OUT; ++x) {
    out[n][y][x] += coeff*in[off+f(0,y,x)]
   }
  }
 }
}
```

Figure 2: Sparse convolution pseudo code. Matrix $\mathbf{W}$ 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 $(0,y,x)$.

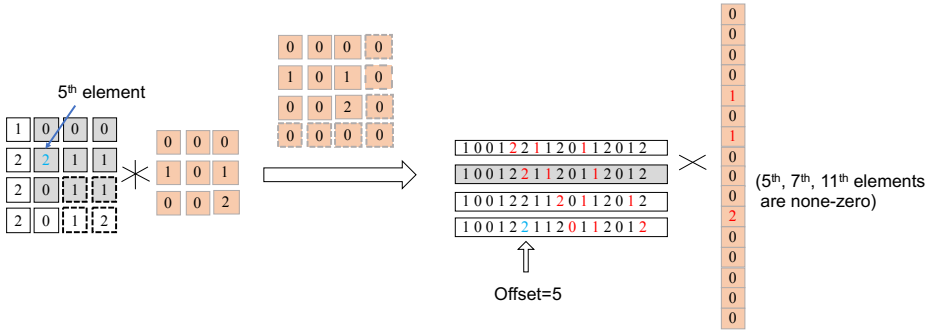
[3] Jongsoo Park et al. (2017). "Faster CNNs with direct sparse convolutions and guided pruning". In: *Proc. ICLR*.

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

Dense Convolution

Sparse Convolution

Reading List

Further Discussion: Reading List

- Baoyuan Liu et al. (2015). “Sparse Convolutional Neural Networks”. In: *Proc. CVPR*, pp. 806–814
- Andrew Lavin and Scott Gray (2016). “Fast Algorithms for Convolutional Neural Networks”. In: *Proc. CVPR*, pp. 4013–4021
- Xingyu Liu et al. (2018). “Efficient sparse-winograd convolutional neural networks”. In: *Proc. ICLR*
- Jason Cong and Bingjun Xiao (2014). “Minimizing computation in convolutional neural networks”. In: *Proc. ICANN*, pp. 281–290