CENG 4480

Embedded System Development & Applications

Lec 02: GEMM

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

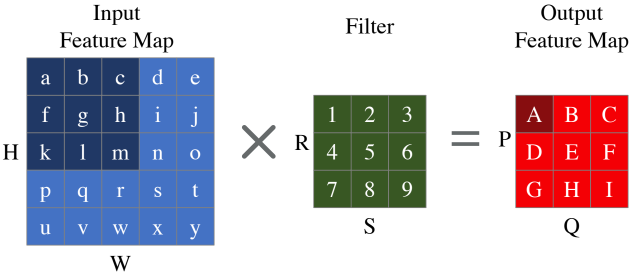
(Latest update: September 23, 2024)

2024 Fall

Overview

Convolution Basis

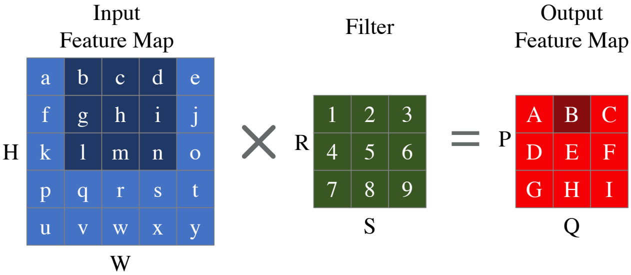
2D-Convolution



$$
\begin{aligned}
A = {} & a \cdot 1 + b \cdot 2 + c \cdot 3 \\
& + f \cdot 4 + g \cdot 5 + h \cdot 6 \\
& + k \cdot 7 + l \cdot 8 + m \cdot 9
\end{aligned}
$$

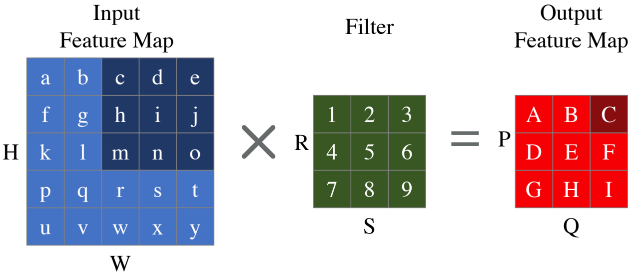
- H: Height of input feature map
- W: Width of input feature map
- R: Height of filter
- S: Width of filter
- P: Height of output feature map
- Q: Width of output feature map

2D-Convolution



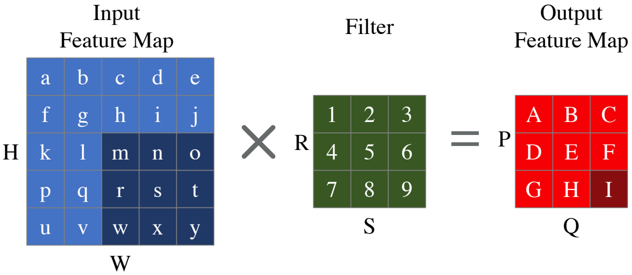
- H: Height of input feature map
- W: Width of input feature map
- R: Height of filter
- S: Width of filter
- P: Height of output feature map
- Q: Width of output feature map
- stride: # of rows/columns traversed per step

2D-Convolution



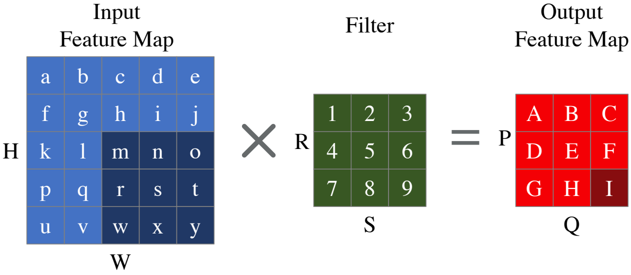
- H: Height of input feature map
- W: Width of input feature map
- R: Height of filter
- S: Width of filter
- P: Height of output feature map
- Q: Width of output feature map
- stride: # of rows/columns traversed per step

2D-Convolution



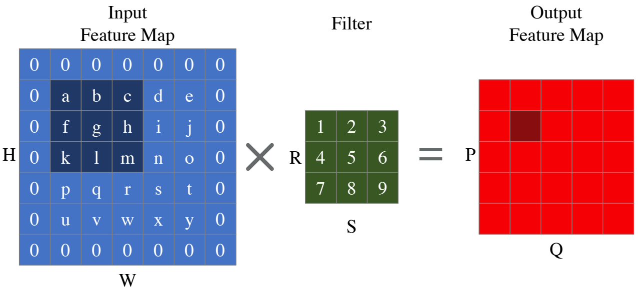
- H: Height of input feature map
- W: Width of input feature map
- R: Height of filter
- S: Width of filter
- P: Height of output feature map
- Q: Width of output feature map
- stride: # of rows/columns traversed per step

2D-Convolution



$$P = \frac{(H - R)}{\text{stride}} + 1;$$
$$Q = \frac{(W - S)}{\text{stride}} + 1.$$

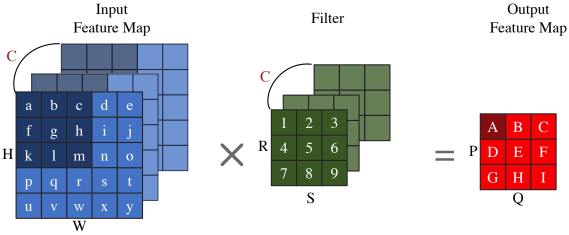
- H: Height of input feature map
- W: Width of input feature map
- R: Height of filter
- S: Width of filter
- P: Height of output feature map
- Q: Width of output feature map
- stride: # of rows/columns traversed per step



$$P = \frac{(H - R + 2 \cdot \text{pad})}{\text{stride}} + 1;$$
$$Q = \frac{(W - S + 2 \cdot \text{pad})}{\text{stride}} + 1.$$

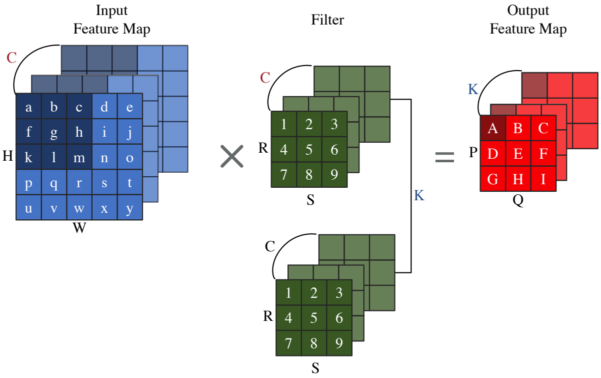
- H: Height of input feature map
- W: Width of input feature map
- R: Height of filter
- S: Width of filter
- P: Height of output feature map
- Q: Width of output feature map
- stride: # of rows/columns traversed per step
- padding: # of zero rows/columns added

3D-Convolution



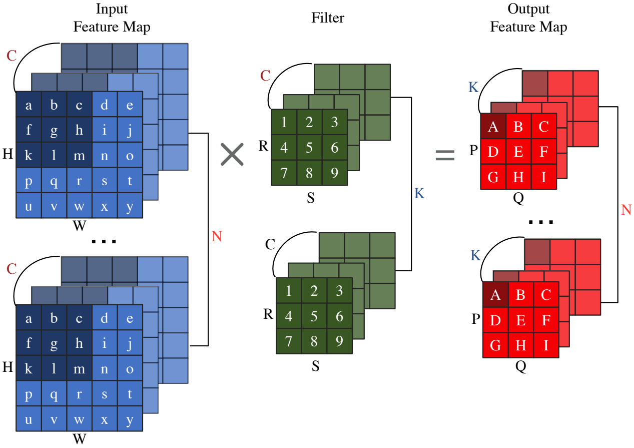
- H: Height of input feature map
- W: Width of input feature map
- R: Height of filter
- S: Width of filter
- P: Height of output feature map
- Q: Width of output feature map
- stride: # of rows/columns traversed per step
- padding: # of zero rows/columns added
- C: # of input channels

3D-Convolution



- H: Height of input feature map
- W: Width of input feature map
- R: Height of filter
- S: Width of filter
- P: Height of output feature map
- Q: Width of output feature map
- stride: # of rows/columns traversed per step
- padding: # of zero rows/columns added
- C: # of input channels
- K: # of output channels

3D-Convolution



- H: Height of input feature map
- W: Width of input feature map
- R: Height of filter
- S: Width of filter
- P: Height of output feature map
- Q: Width of output feature map
- stride: # of rows/columns traversed per step
- padding: # of zero rows/columns added
- C: # of input channels
- K: # of output channels
- N: Batch size

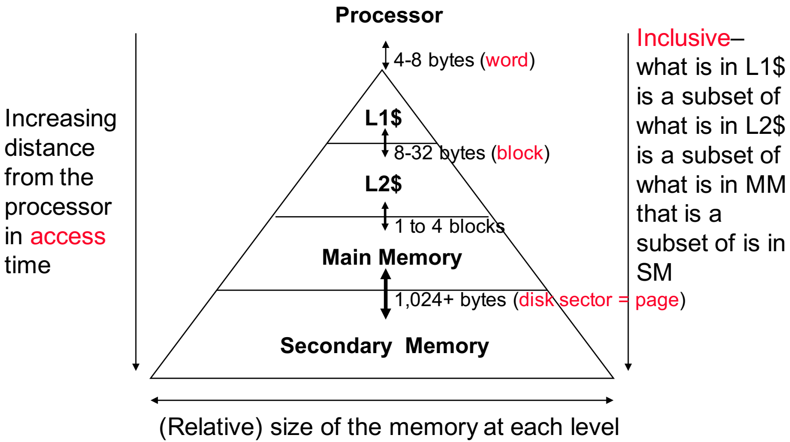
Convolution 101

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2 & 2 & 1 & 1 & 2 & 0 \\
0 & 2 & 0 & 1 & 1 & 0 & 0 \\
0 & 2 & 0 & 1 & 2 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
*
\begin{bmatrix}
1 & 0 & 0 \\
1 & 1 & 1 \\
1 & 0 & -1
\end{bmatrix}
=
\begin{bmatrix}
4 & 6 & 3 & 5 & 4 \\
2 & 6 & 2 & 4 & 4 \\
1 & 5 & 3 & 4 & 4 \\
2 & 4 & 3 & 3 & 4 \\
0 & 2 & 2 & 4 & 3
\end{bmatrix}
$$

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

Why the standard SRAM chips are costly?

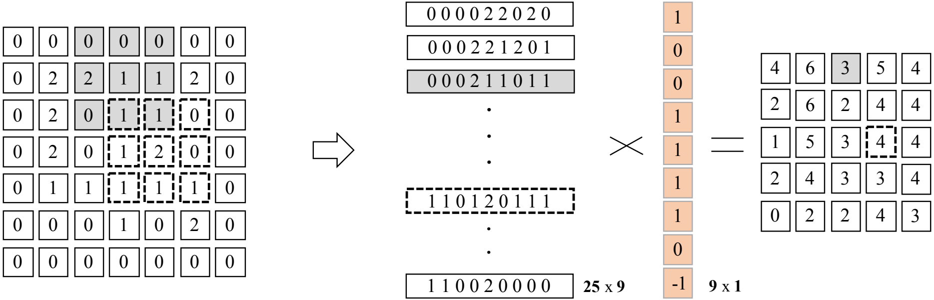
- A: They use highly advanced micro-electronic devices
- B: They house 6 transistor per chip
- C: They require specially designed PCB's
- D: None of the mentioned

The fastest data access is?

- A: Caches
- B: DRAM's
- C: SRAM's
- D: Registers

Im2Col

Im2col (Image2Column) Convolution

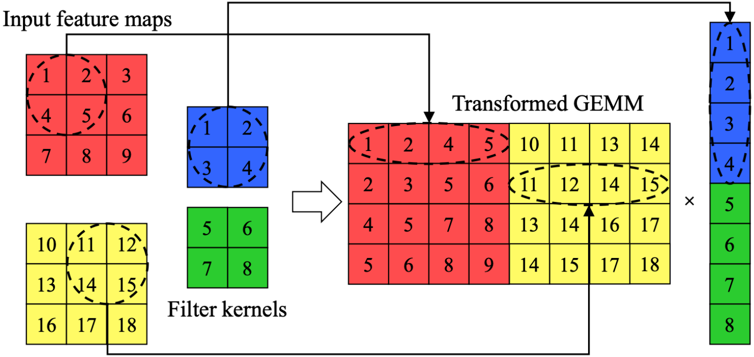


- Large extra memory overhead
- Good performance
- BLAS-friendly memory layout to enjoy SIMD/locality/parallelism
- Applicable for any convolution configuration on any platform

For an $n \times n$ kernels, at most how many times an input feature map value will be duplicated?

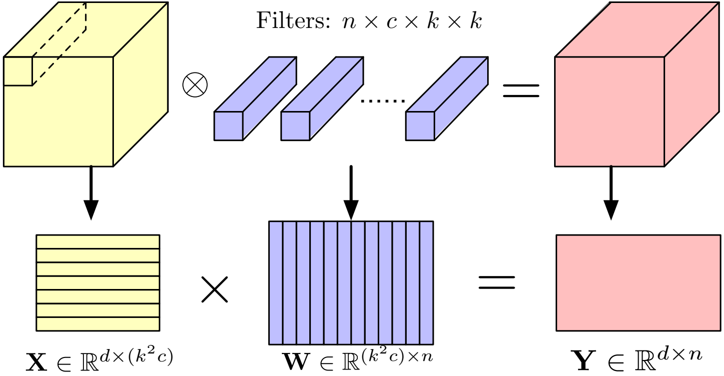
- A: n
- B: $2n$
- C: n^2
- D: n^3

`Im2col` (Image2Column) Convolution



- Input channel #: 2
- Output channel #: 1

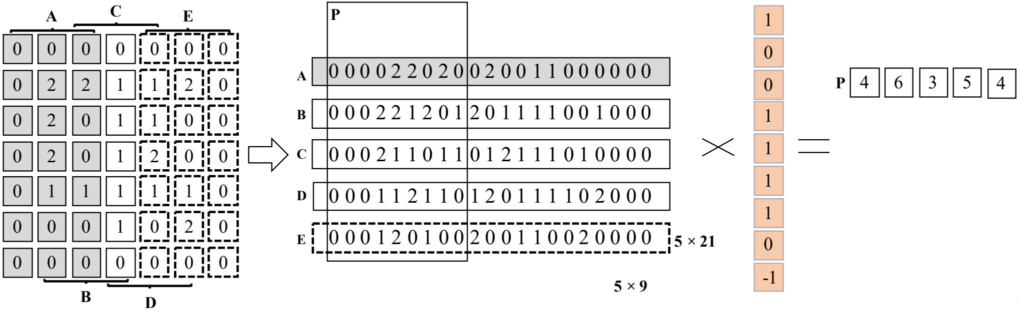
Im2col (Image2Column) Convolution



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

Memory-efficient Convolution

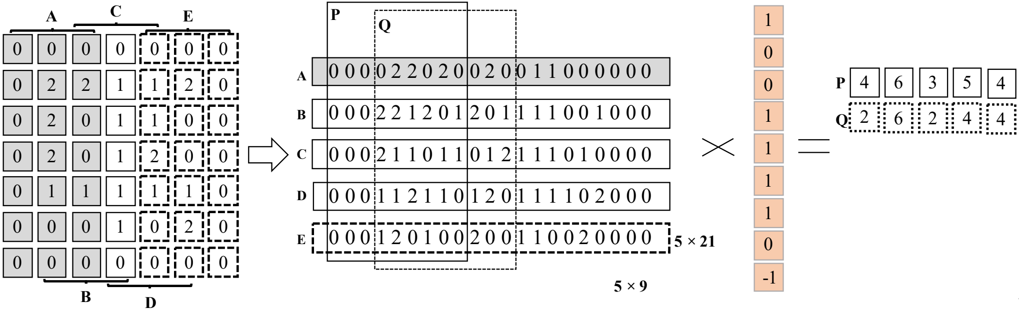
Memory-efficient Convolution (MEC)



[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*.

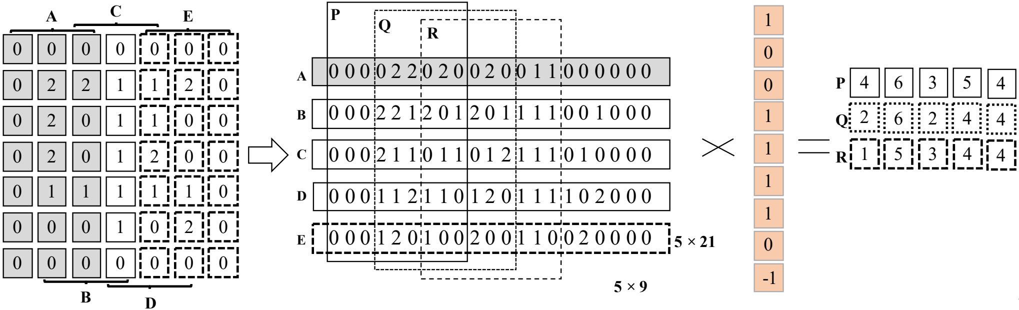


[1]

- 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*.

Memory-efficient Convolution (MEC)

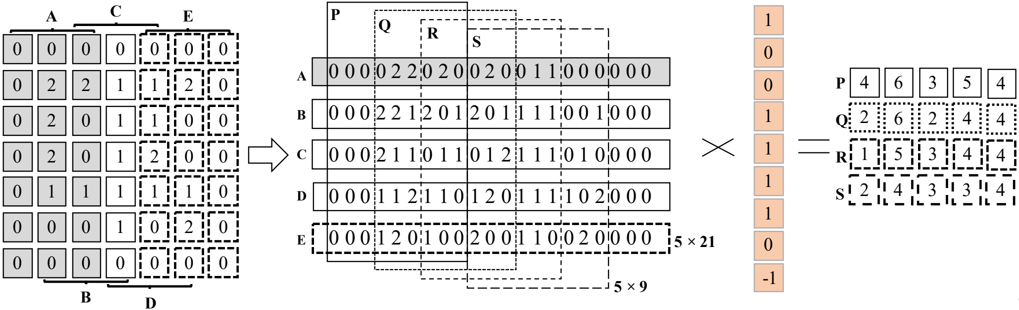


[1]

- 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*.

Memory-efficient Convolution (MEC)

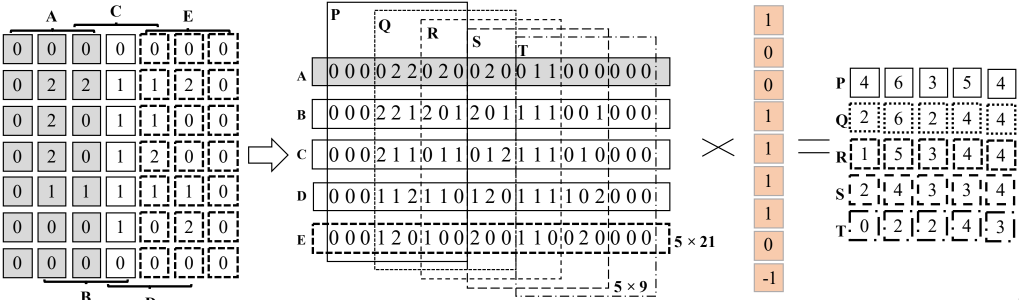


[1]

- 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*.

Memory-efficient Convolution (MEC)



[1]

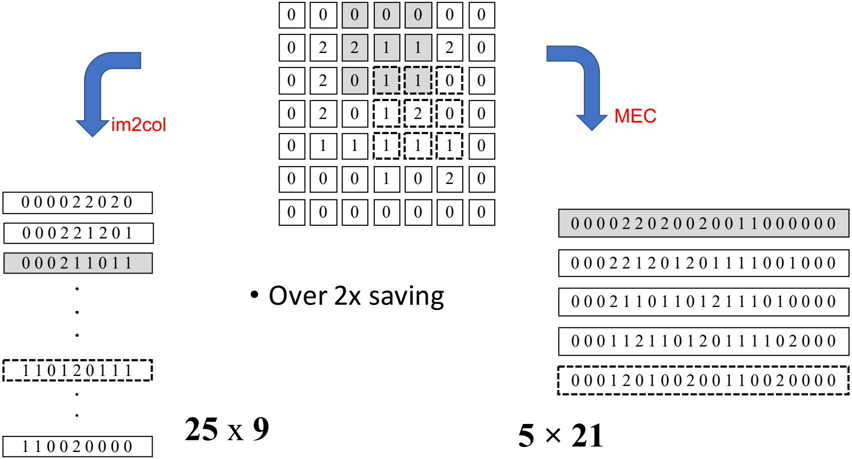
- Sub matrices in the lowered matrix will be "sgemm" ed in parallel
- Smaller memory foot print, cache locality, and explicit parallelism

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

For an $n \times n$ kernels, at most how many times an input feature map value will be duplicated?

- A: n
- B: $2n$
- C: n^2
- D: n^3

Over 2× memory saving[2]:



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

Memory Layout

Data Layout Formats[3]

- N is the batch size
- C is the number of feature maps
- H is the image height
- W is the image width

EXAMPLE

N = 1

C = 64

H = 5

W = 4

c = 0

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19

c = 1

20	21	22	23
24	25	26	27
28	29	30	31
32	33	34	35
36	37	38	39

c = 2

40	41	42	43
44	45	46	47
48	49	50	51
52	53	54	55
56	57	58	59

. . .

c = 30

600	601	602	603
604	605	606	607
608	609	610	611
612	613	614	615
616	617	618	619

c = 31

620	621	622	623
624	625	626	627
628	629	630	631
632	633	634	635
636	637	638	639

c = 32

640	641	642	643
644	645	646	647
648	649	650	651
652	653	654	655
656	657	658	659

. . .

. . .

c = 62

1240	1241	1242	1243
1244	1245	1246	1247
1248	1249	1250	1251
1252	1253	1254	1255
1256	1257	1258	1259

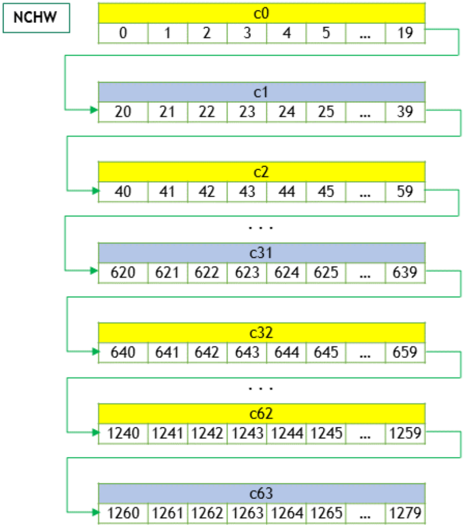
c = 63

1260	1261	1262	1263
1264	1265	1266	1267
1268	1269	1270	1271
1272	1273	1274	1275
1276	1277	1278	1279

[3] https://docs.nvidia.com/deeplearning/cudnn/developer-guide/index.html

NCHW Memory Layout

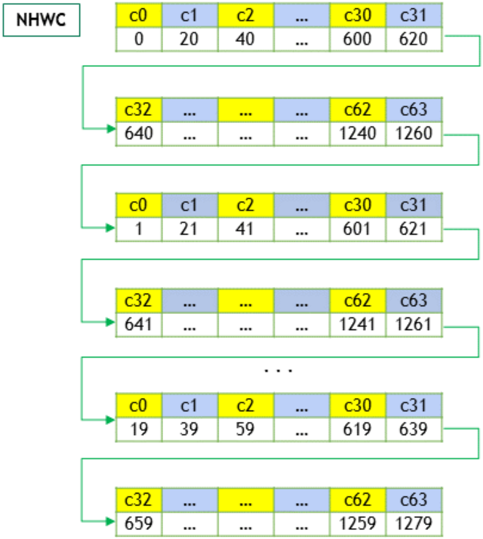
- Begin with first channel (c=0), elements arranged contiguously in row-major order
- Continue with second and subsequent channels until all channels are laid out



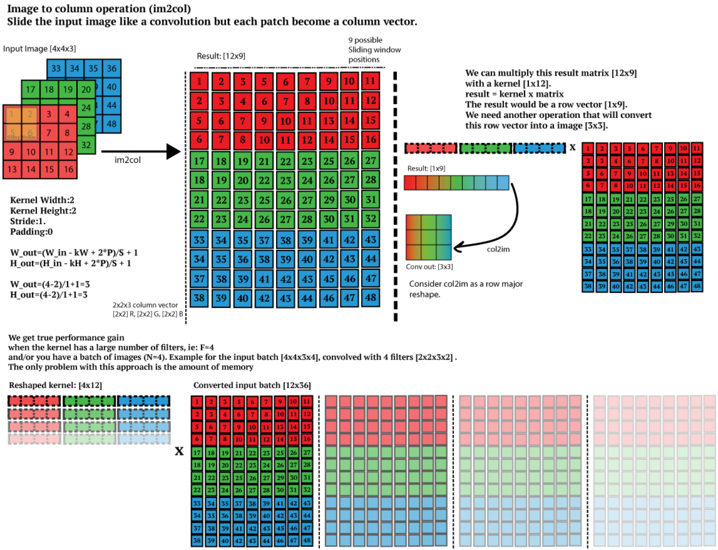
NHWC Memory Layout

- Begin with the first element of channel 0, then proceed to the first element of channel 1, and so on, until the first elements of all the C channels are laid out
- Next, select the second element of channel 0, then proceed to the second element of channel 1, and so on, until the second element of all the channels are laid out
- Follow the row-major order of channel 0 and complete all the elements
- Proceed to the next batch (if N is > 1)

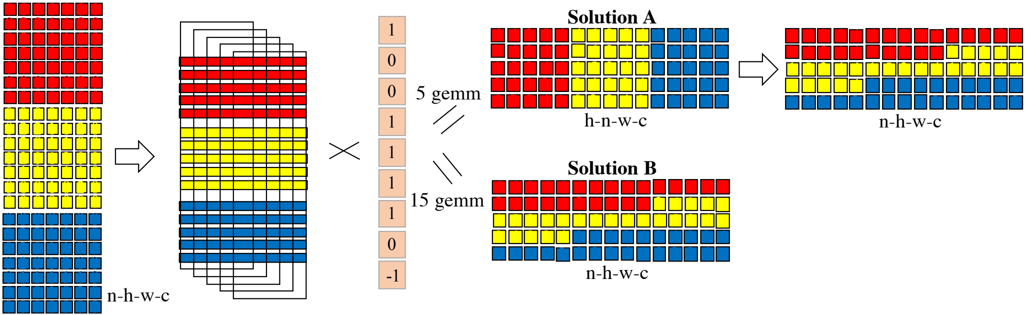
NHWC Memory Layout



Memory Layout in `Im2col`

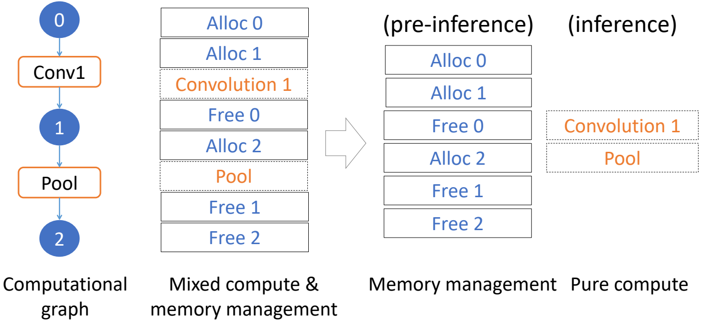


[4] https://leonardoaraujosantos.gitbook.io/artificial-inteligence/machine_learning/deep_learning/convolution_layer/making_faster



- MEC w. mini-batch: can use `n-h-w-c` format
- Fusing convolution+pooling can be another solution

Memory optimization of MNN



- MNN can infer the exact required memory for the entire graph:
  - virtually walking through all operations
  - summing up all allocation and freeing